

libstdc++

Generated by Doxygen 1.8.15

---

<b>1 Mathematical Special Functions</b>	<b>1</b>
1.1 Introduction and History	2
1.2 Contents	2
1.3 General Features	3
1.3.1 Argument Promotion	3
1.3.2 NaN Arguments	3
1.4 Implementation	3
1.5 Testing	3
1.6 General Bibliography	3
<b>2 Todo List</b>	<b>4</b>
<b>3 Module Documentation</b>	<b>5</b>
3.1 Adaptors for pointers to functions	5
3.1.1 Detailed Description	6
3.1.2 Function Documentation	6
3.2 Adaptors for pointers to members	7
3.2.1 Detailed Description	7
3.3 Algorithms	8
3.3.1 Detailed Description	8
3.4 Allocators	9
3.4.1 Detailed Description	10
3.4.2 Typedef Documentation	10
3.5 Arithmetic Classes	11
3.5.1 Detailed Description	11
3.6 Array creation functions	12
3.6.1 Detailed Description	12
3.7 Associative	13
3.7.1 Detailed Description	13
3.8 Atomics	14
3.8.1 Detailed Description	18
3.8.2 Macro Definition Documentation	18
3.8.3 Typedef Documentation	19
3.8.4 Enumeration Type Documentation	28
3.8.5 Function Documentation	28
3.9 Base and Implementation Classes	29
3.9.1 Detailed Description	30
3.9.2 Enumeration Type Documentation	30
3.10 Base and Implementation Classes	31
3.10.1 Detailed Description	33

3.10.2 Function Documentation . . . . .	33
3.11 Base and Policy Classes . . . . .	34
3.11.1 Detailed Description . . . . .	34
3.12 Base and Policy Classes . . . . .	35
3.12.1 Detailed Description . . . . .	35
3.13 Base and Policy Classes . . . . .	36
3.13.1 Detailed Description . . . . .	36
3.14 Bernoulli Distributions . . . . .	37
3.14.1 Detailed Description . . . . .	37
3.14.2 Function Documentation . . . . .	37
3.15 Binary Search . . . . .	41
3.15.1 Detailed Description . . . . .	41
3.15.2 Function Documentation . . . . .	42
3.16 Binder Classes . . . . .	47
3.16.1 Detailed Description . . . . .	48
3.16.2 Function Documentation . . . . .	48
3.17 Boolean Operations Classes . . . . .	50
3.17.1 Detailed Description . . . . .	50
3.18 Branch-Based . . . . .	51
3.18.1 Detailed Description . . . . .	51
3.19 Comparison Classes . . . . .	52
3.19.1 Detailed Description . . . . .	52
3.20 Complex Numbers . . . . .	53
3.20.1 Detailed Description . . . . .	56
3.20.2 Function Documentation . . . . .	56
3.21 Concurrency . . . . .	70
3.21.1 Detailed Description . . . . .	70
3.22 Condition Variables . . . . .	71
3.22.1 Detailed Description . . . . .	71
3.22.2 Enumeration Type Documentation . . . . .	71
3.23 Const-propagating wrapper . . . . .	72
3.23.1 Detailed Description . . . . .	73
3.24 Containers . . . . .	74
3.24.1 Detailed Description . . . . .	74
3.25 Containers . . . . .	75
3.25.1 Detailed Description . . . . .	75
3.26 Data Structure Type . . . . .	76
3.26.1 Detailed Description . . . . .	76
3.27 Decimal Floating-Point Arithmetic . . . . .	77

3.27.1 Detailed Description . . . . .	77
3.28 Diagnostics . . . . .	78
3.28.1 Detailed Description . . . . .	78
3.29 Dynamic Bitset. . . . .	79
3.29.1 Detailed Description . . . . .	80
3.29.2 Function Documentation . . . . .	80
3.30 Exceptions . . . . .	84
3.30.1 Detailed Description . . . . .	84
3.31 Exceptions . . . . .	85
3.31.1 Detailed Description . . . . .	86
3.31.2 Function Documentation . . . . .	86
3.32 Experimental . . . . .	88
3.32.1 Detailed Description . . . . .	88
3.33 Extensions . . . . .	89
3.33.1 Detailed Description . . . . .	89
3.34 Filesystem TS . . . . .	90
3.34.1 Detailed Description . . . . .	93
3.34.2 Enumeration Type Documentation . . . . .	94
3.35 Function Objects . . . . .	95
3.35.1 Detailed Description . . . . .	96
3.35.2 Function Documentation . . . . .	96
3.36 Futures . . . . .	97
3.36.1 Detailed Description . . . . .	98
3.36.2 Enumeration Type Documentation . . . . .	98
3.36.3 Function Documentation . . . . .	99
3.37 Hash-Based . . . . .	101
3.37.1 Detailed Description . . . . .	101
3.38 Hashes . . . . .	102
3.38.1 Detailed Description . . . . .	102
3.39 Heap . . . . .	103
3.39.1 Detailed Description . . . . .	103
3.39.2 Function Documentation . . . . .	103
3.40 Heap-Based . . . . .	110
3.40.1 Detailed Description . . . . .	111
3.40.2 Function Documentation . . . . .	111
3.41 I/O . . . . .	113
3.41.1 Detailed Description . . . . .	114
3.41.2 Typedef Documentation . . . . .	114
3.42 Invalidation Guarantees . . . . .	120



3.42.1 Detailed Description	120
3.43 Iterator Tags	121
3.43.1 Detailed Description	121
3.44 Iterators	122
3.44.1 Detailed Description	125
3.44.2 Function Documentation	125
3.45 List-Based	129
3.45.1 Detailed Description	129
3.46 Locales	130
3.46.1 Detailed Description	130
3.46.2 Function Documentation	131
3.47 Mathematical Special Functions	133
3.47.1 Detailed Description	135
3.47.2 Function Documentation	135
3.48 Mathematical Special Functions	164
3.48.1 Detailed Description	166
3.48.2 Function Documentation	166
3.49 Memory	172
3.49.1 Detailed Description	172
3.50 Metaprogramming	173
3.50.1 Detailed Description	176
3.50.2 Typedef Documentation	177
3.50.3 Variable Documentation	182
3.51 Mutating	183
3.51.1 Detailed Description	185
3.51.2 Function Documentation	185
3.52 Mutexes	207
3.52.1 Detailed Description	207
3.52.2 Macro Definition Documentation	207
3.52.3 Typedef Documentation	208
3.52.4 Function Documentation	208
3.52.5 Variable Documentation	209
3.53 Negators	210
3.53.1 Detailed Description	210
3.53.2 Function Documentation	210
3.54 Non-Mutating	212
3.54.1 Detailed Description	213
3.54.2 Function Documentation	213
3.55 Normal Distributions	233

3.55.1 Detailed Description . . . . .	234
3.55.2 Function Documentation . . . . .	234
3.56 Numeric Arrays . . . . .	237
3.56.1 Detailed Description . . . . .	245
3.56.2 Function Documentation . . . . .	245
3.57 Numerics . . . . .	283
3.57.1 Detailed Description . . . . .	283
3.58 Optional values . . . . .	284
3.58.1 Detailed Description . . . . .	287
3.58.2 Function Documentation . . . . .	287
3.58.3 Variable Documentation . . . . .	287
3.59 Pointer Abstractions . . . . .	288
3.59.1 Detailed Description . . . . .	291
3.59.2 Function Documentation . . . . .	292
3.60 Poisson Distributions . . . . .	307
3.60.1 Detailed Description . . . . .	308
3.60.2 Function Documentation . . . . .	308
3.61 Policy-Based Data Structures . . . . .	314
3.61.1 Detailed Description . . . . .	314
3.62 Random Number Distributions . . . . .	315
3.62.1 Detailed Description . . . . .	315
3.63 Random Number Generation . . . . .	316
3.63.1 Detailed Description . . . . .	316
3.63.2 Function Documentation . . . . .	316
3.64 Random Number Generators . . . . .	317
3.64.1 Detailed Description . . . . .	318
3.64.2 Typedef Documentation . . . . .	318
3.64.3 Function Documentation . . . . .	319
3.65 Random Number Utilities . . . . .	323
3.65.1 Detailed Description . . . . .	323
3.66 Rational Arithmetic . . . . .	324
3.66.1 Detailed Description . . . . .	325
3.66.2 Typedef Documentation . . . . .	325
3.67 Regular Expressions . . . . .	326
3.67.1 Detailed Description . . . . .	331
3.67.2 Typedef Documentation . . . . .	331
3.67.3 Function Documentation . . . . .	333
3.68 SGI . . . . .	372
3.68.1 Detailed Description . . . . .	373

---

3.68.2 Function Documentation . . . . .	374
3.69 Sequences . . . . .	384
3.69.1 Detailed Description . . . . .	384
3.70 Set Operation . . . . .	385
3.70.1 Detailed Description . . . . .	385
3.70.2 Function Documentation . . . . .	386
3.71 Sorting . . . . .	394
3.71.1 Detailed Description . . . . .	396
3.71.2 Function Documentation . . . . .	396
3.72 Strings . . . . .	417
3.72.1 Detailed Description . . . . .	417
3.72.2 Typedef Documentation . . . . .	417
3.73 Tags . . . . .	419
3.73.1 Detailed Description . . . . .	419
3.73.2 Typedef Documentation . . . . .	419
3.74 Threads . . . . .	420
3.74.1 Detailed Description . . . . .	420
3.75 Time . . . . .	421
3.75.1 Detailed Description . . . . .	421
3.76 Traits . . . . .	422
3.76.1 Detailed Description . . . . .	423
3.77 Type-safe container of any type . . . . .	424
3.77.1 Detailed Description . . . . .	424
3.77.2 Function Documentation . . . . .	425
3.78 Uniform Distributions . . . . .	429
3.78.1 Detailed Description . . . . .	429
3.78.2 Function Documentation . . . . .	429
3.79 Unordered Associative . . . . .	433
3.79.1 Detailed Description . . . . .	433
3.80 Utilities . . . . .	434
3.80.1 Detailed Description . . . . .	437
3.80.2 Function Documentation . . . . .	437
3.80.3 Variable Documentation . . . . .	446
<b>4 Namespace Documentation . . . . .</b>	<b>447</b>
4.1 <code>__gnu_cxx</code> Namespace Reference . . . . .	447
4.1.1 Detailed Description . . . . .	462
4.1.2 Function Documentation . . . . .	462
4.2 <code>__gnu_cxx::__detail</code> Namespace Reference . . . . .	479

---

4.2.1 Detailed Description . . . . .	480
4.2.2 Function Documentation . . . . .	480
4.3 <code>__gnu_cxx::typelist</code> Namespace Reference . . . . .	481
4.3.1 Detailed Description . . . . .	481
4.3.2 Function Documentation . . . . .	482
4.4 <code>__gnu_debug</code> Namespace Reference . . . . .	482
4.4.1 Detailed Description . . . . .	489
4.4.2 Enumeration Type Documentation . . . . .	489
4.4.3 Function Documentation . . . . .	489
4.5 <code>__gnu_internal</code> Namespace Reference . . . . .	494
4.5.1 Detailed Description . . . . .	494
4.6 <code>__gnu_parallel</code> Namespace Reference . . . . .	495
4.6.1 Detailed Description . . . . .	502
4.6.2 Typedef Documentation . . . . .	503
4.6.3 Enumeration Type Documentation . . . . .	503
4.6.4 Function Documentation . . . . .	505
4.6.5 Variable Documentation . . . . .	554
4.7 <code>__gnu_pbds</code> Namespace Reference . . . . .	555
4.7.1 Detailed Description . . . . .	557
4.8 <code>__gnu_profile</code> Namespace Reference . . . . .	557
4.8.1 Detailed Description . . . . .	561
4.8.2 Typedef Documentation . . . . .	561
4.8.3 Function Documentation . . . . .	561
4.9 <code>__gnu_sequential</code> Namespace Reference . . . . .	562
4.9.1 Detailed Description . . . . .	562
4.10 <code>abi</code> Namespace Reference . . . . .	562
4.10.1 Detailed Description . . . . .	562
4.11 <code>std</code> Namespace Reference . . . . .	562
4.11.1 Detailed Description . . . . .	669
4.11.2 Typedef Documentation . . . . .	669
4.11.3 Enumeration Type Documentation . . . . .	672
4.11.4 Function Documentation . . . . .	673
4.11.5 Variable Documentation . . . . .	788
4.12 <code>std::__debug</code> Namespace Reference . . . . .	792
4.12.1 Detailed Description . . . . .	796
4.12.2 Function Documentation . . . . .	796
4.13 <code>std::__detail</code> Namespace Reference . . . . .	797
4.13.1 Detailed Description . . . . .	800
4.13.2 Function Documentation . . . . .	800

4.14	<a href="#">std::__parallel Namespace Reference</a>	801
4.14.1	<a href="#">Detailed Description</a>	818
4.15	<a href="#">std::__profile Namespace Reference</a>	819
4.15.1	<a href="#">Detailed Description</a>	824
4.15.2	<a href="#">Function Documentation</a>	824
4.16	<a href="#">std::chrono Namespace Reference</a>	825
4.16.1	<a href="#">Detailed Description</a>	827
4.16.2	<a href="#">Typedef Documentation</a>	827
4.16.3	<a href="#">Function Documentation</a>	828
4.17	<a href="#">std::decimal Namespace Reference</a>	829
4.17.1	<a href="#">Detailed Description</a>	838
4.17.2	<a href="#">Function Documentation</a>	838
4.18	<a href="#">std::placeholders Namespace Reference</a>	838
4.18.1	<a href="#">Detailed Description</a>	839
4.19	<a href="#">std::regex_constants Namespace Reference</a>	839
4.19.1	<a href="#">Detailed Description</a>	840
4.19.2	<a href="#">Enumeration Type Documentation</a>	840
4.19.3	<a href="#">Function Documentation</a>	842
4.19.4	<a href="#">Variable Documentation</a>	848
4.20	<a href="#">std::rel_ops Namespace Reference</a>	854
4.20.1	<a href="#">Detailed Description</a>	854
4.20.2	<a href="#">Function Documentation</a>	855
4.21	<a href="#">std::this_thread Namespace Reference</a>	857
4.21.1	<a href="#">Detailed Description</a>	858
4.21.2	<a href="#">Function Documentation</a>	858
4.22	<a href="#">std::tr1 Namespace Reference</a>	859
4.22.1	<a href="#">Detailed Description</a>	861
4.22.2	<a href="#">Function Documentation</a>	861
4.23	<a href="#">std::tr1::__detail Namespace Reference</a>	862
4.23.1	<a href="#">Detailed Description</a>	862
4.24	<a href="#">std::tr2 Namespace Reference</a>	862
4.24.1	<a href="#">Detailed Description</a>	864
4.25	<a href="#">std::tr2::__detail Namespace Reference</a>	864
4.25.1	<a href="#">Detailed Description</a>	864
<b>5</b>	<b>Class Documentation</b>	<b>864</b>
5.1	<a href="#">__cxxabiv1::__forced_unwind Class Reference</a>	864
5.1.1	<a href="#">Detailed Description</a>	864
5.2	<a href="#">__gnu_cxx::__alloc_traits&lt; _Alloc, typename &gt; Struct Template Reference</a>	865

5.2.1 Detailed Description	866
5.2.2 Member Typedef Documentation	867
5.2.3 Member Function Documentation	868
5.3 <code>__gnu_cxx::__common_pool_policy&lt;_PoolTp, _Thread&gt;</code> Struct Template Reference	875
5.3.1 Detailed Description	875
5.4 <code>__gnu_cxx::__detail::__mini_vector&lt;_Tp&gt;</code> Class Template Reference	875
5.4.1 Detailed Description	876
5.5 <code>__gnu_cxx::__detail::__Bitmap_counter&lt;_Tp&gt;</code> Class Template Reference	876
5.5.1 Detailed Description	876
5.6 <code>__gnu_cxx::__detail::__Ffit_finder&lt;_Tp&gt;</code> Class Template Reference	877
5.6.1 Detailed Description	877
5.6.2 Member Typedef Documentation	877
5.7 <code>__gnu_cxx::__mt_alloc&lt;_Tp, _Poolp&gt;</code> Class Template Reference	878
5.7.1 Detailed Description	879
5.8 <code>__gnu_cxx::__mt_alloc_base&lt;_Tp&gt;</code> Class Template Reference	879
5.8.1 Detailed Description	880
5.9 <code>__gnu_cxx::__per_type_pool_policy&lt;_Tp, _PoolTp, _Thread&gt;</code> Struct Template Reference	880
5.9.1 Detailed Description	880
5.10 <code>__gnu_cxx::__pool&lt;_Thread&gt;</code> Class Template Reference	881
5.10.1 Detailed Description	881
5.11 <code>__gnu_cxx::__pool&lt;false&gt;</code> Class Template Reference	881
5.11.1 Detailed Description	882
5.12 <code>__gnu_cxx::__pool&lt;true&gt;</code> Class Template Reference	882
5.12.1 Detailed Description	883
5.13 <code>__gnu_cxx::__pool_alloc&lt;_Tp&gt;</code> Class Template Reference	883
5.13.1 Detailed Description	885
5.14 <code>__gnu_cxx::__pool_alloc_base</code> Class Reference	885
5.14.1 Detailed Description	886
5.15 <code>__gnu_cxx::__pool_base</code> Struct Reference	886
5.15.1 Detailed Description	887
5.16 <code>__gnu_cxx::__rc_string_base&lt;_CharT, _Traits, _Alloc&gt;</code> Class Template Reference	887
5.16.1 Detailed Description	889
5.17 <code>__gnu_cxx::__scoped_lock</code> Class Reference	890
5.17.1 Detailed Description	890
5.18 <code>__gnu_cxx::__versa_string&lt;_CharT, _Traits, _Alloc, _Base&gt;</code> Class Template Reference	890
5.18.1 Detailed Description	894
5.18.2 Constructor & Destructor Documentation	894
5.18.3 Member Function Documentation	898
5.18.4 Member Data Documentation	959

5.19	<a href="#">__gnu_cxx::_Caster&lt;_ToType&gt; Struct Template Reference</a>	960
5.19.1	Detailed Description	960
5.20	<a href="#">__gnu_cxx::_Char_types&lt;_CharT&gt; Struct Template Reference</a>	960
5.20.1	Detailed Description	961
5.21	<a href="#">__gnu_cxx::_ExtPtr_allocator&lt;_Tp&gt; Class Template Reference</a>	961
5.21.1	Detailed Description	962
5.22	<a href="#">__gnu_cxx::_Invalid_type Struct Reference</a>	962
5.22.1	Detailed Description	962
5.23	<a href="#">__gnu_cxx::_Pointer_adapter&lt;_Storage_policy&gt; Class Template Reference</a>	962
5.23.1	Detailed Description	964
5.24	<a href="#">__gnu_cxx::_Relative_pointer_impl&lt;_Tp&gt; Class Template Reference</a>	965
5.24.1	Detailed Description	965
5.25	<a href="#">__gnu_cxx::_Relative_pointer_impl&lt;const _Tp&gt; Class Template Reference</a>	965
5.25.1	Detailed Description	966
5.26	<a href="#">__gnu_cxx::_Std_pointer_impl&lt;_Tp&gt; Class Template Reference</a>	966
5.26.1	Detailed Description	966
5.27	<a href="#">__gnu_cxx::_Unqualified_type&lt;_Tp&gt; Struct Template Reference</a>	967
5.27.1	Detailed Description	967
5.28	<a href="#">__gnu_cxx::annotate_base Struct Reference</a>	967
5.28.1	Detailed Description	968
5.29	<a href="#">__gnu_cxx::array_allocator&lt;_Tp, _Array&gt; Class Template Reference</a>	968
5.29.1	Detailed Description	969
5.30	<a href="#">__gnu_cxx::array_allocator_base&lt;_Tp&gt; Class Template Reference</a>	969
5.30.1	Detailed Description	970
5.31	<a href="#">__gnu_cxx::binary_compose&lt;_Operation1, _Operation2, _Operation3&gt; Class Template Reference</a>	970
5.31.1	Detailed Description	971
5.31.2	Member Typedef Documentation	971
5.32	<a href="#">__gnu_cxx::bitmap_allocator&lt;_Tp&gt; Class Template Reference</a>	972
5.32.1	Detailed Description	973
5.32.2	Member Function Documentation	973
5.33	<a href="#">__gnu_cxx::char_traits&lt;_CharT&gt; Struct Template Reference</a>	974
5.33.1	Detailed Description	975
5.34	<a href="#">__gnu_cxx::character&lt;_Value, _Int, _St&gt; Struct Template Reference</a>	976
5.34.1	Detailed Description	976
5.35	<a href="#">__gnu_cxx::condition_base Struct Reference</a>	976
5.35.1	Detailed Description	977
5.36	<a href="#">__gnu_cxx::constant_binary_fun&lt;_Result, _Arg1, _Arg2&gt; Struct Template Reference</a>	977
5.36.1	Detailed Description	977
5.37	<a href="#">__gnu_cxx::constant_unary_fun&lt;_Result, _Argument&gt; Struct Template Reference</a>	978

5.37.1 Detailed Description	978
5.38 <code>__gnu_cxx::constant_void_fun&lt;_Result&gt;</code> Struct Template Reference	978
5.38.1 Detailed Description	979
5.39 <code>__gnu_cxx::debug_allocator&lt;_Alloc&gt;</code> Class Template Reference	979
5.39.1 Detailed Description	980
5.40 <code>__gnu_cxx::enc_filebuf&lt;_CharT&gt;</code> Class Template Reference	980
5.40.1 Detailed Description	983
5.40.2 Member Function Documentation	983
5.40.3 Member Data Documentation	1003
5.41 <code>__gnu_cxx::encoding_char_traits&lt;_CharT&gt;</code> Struct Template Reference	1007
5.41.1 Detailed Description	1008
5.42 <code>__gnu_cxx::encoding_state</code> Class Reference	1008
5.42.1 Detailed Description	1009
5.43 <code>__gnu_cxx::forced_error</code> Struct Reference	1009
5.43.1 Detailed Description	1010
5.44 <code>__gnu_cxx::free_list</code> Class Reference	1010
5.44.1 Detailed Description	1010
5.44.2 Member Function Documentation	1011
5.45 <code>__gnu_cxx::hash_map&lt;_Key, _Tp, _HashFn, _EqualKey, _Alloc&gt;</code> Class Template Reference	1012
5.45.1 Detailed Description	1013
5.46 <code>__gnu_cxx::hash_multimap&lt;_Key, _Tp, _HashFn, _EqualKey, _Alloc&gt;</code> Class Template Reference	1014
5.46.1 Detailed Description	1015
5.47 <code>__gnu_cxx::hash_multiset&lt;_Value, _HashFcn, _EqualKey, _Alloc&gt;</code> Class Template Reference	1015
5.47.1 Detailed Description	1017
5.48 <code>__gnu_cxx::hash_set&lt;_Value, _HashFcn, _EqualKey, _Alloc&gt;</code> Class Template Reference	1017
5.48.1 Detailed Description	1018
5.49 <code>__gnu_cxx::limit_condition</code> Struct Reference	1019
5.49.1 Detailed Description	1019
5.50 <code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference	1019
5.50.1 Detailed Description	1019
5.51 <code>__gnu_cxx::limit_condition::limit_adjustor</code> Struct Reference	1020
5.51.1 Detailed Description	1020
5.52 <code>__gnu_cxx::limit_condition::never_adjustor</code> Struct Reference	1020
5.52.1 Detailed Description	1020
5.53 <code>__gnu_cxx::malloc_allocator&lt;_Tp&gt;</code> Class Template Reference	1020
5.53.1 Detailed Description	1021
5.54 <code>__gnu_cxx::new_allocator&lt;_Tp&gt;</code> Class Template Reference	1021
5.54.1 Detailed Description	1022
5.55 <code>__gnu_cxx::project1st&lt;_Arg1, _Arg2&gt;</code> Struct Template Reference	1023



5.55.1 Detailed Description	1023
5.55.2 Member Typedef Documentation	1023
5.56 <a href="#">__gnu_cxx::project2nd&lt;_Arg1, _Arg2&gt; Struct Template Reference</a>	1024
5.56.1 Detailed Description	1024
5.56.2 Member Typedef Documentation	1024
5.57 <a href="#">__gnu_cxx::random_condition Struct Reference</a>	1025
5.57.1 Detailed Description	1026
5.58 <a href="#">__gnu_cxx::random_condition::always_adjustor Struct Reference</a>	1026
5.58.1 Detailed Description	1026
5.59 <a href="#">__gnu_cxx::random_condition::group_adjustor Struct Reference</a>	1026
5.59.1 Detailed Description	1027
5.60 <a href="#">__gnu_cxx::random_condition::never_adjustor Struct Reference</a>	1027
5.60.1 Detailed Description	1027
5.61 <a href="#">__gnu_cxx::rb_tree&lt;_Key, _Value, _KeyOfValue, _Compare, _Alloc&gt; Struct Template Reference</a>	1027
5.61.1 Detailed Description	1031
5.62 <a href="#">__gnu_cxx::recursive_init_error Class Reference</a>	1031
5.62.1 Detailed Description	1031
5.63 <a href="#">__gnu_cxx::rope&lt;_CharT, _Alloc&gt; Class Template Reference</a>	1032
5.63.1 Detailed Description	1036
5.64 <a href="#">__gnu_cxx::select1st&lt;_Pair&gt; Struct Template Reference</a>	1037
5.64.1 Detailed Description	1037
5.64.2 Member Typedef Documentation	1037
5.65 <a href="#">__gnu_cxx::select2nd&lt;_Pair&gt; Struct Template Reference</a>	1038
5.65.1 Detailed Description	1038
5.65.2 Member Typedef Documentation	1038
5.66 <a href="#">__gnu_cxx::slist&lt;_Tp, _Alloc&gt; Class Template Reference</a>	1039
5.66.1 Detailed Description	1041
5.67 <a href="#">__gnu_cxx::stdio_filebuf&lt;_CharT, _Traits&gt; Class Template Reference</a>	1042
5.67.1 Detailed Description	1045
5.67.2 Constructor & Destructor Documentation	1045
5.67.3 Member Function Documentation	1046
5.67.4 Member Data Documentation	1067
5.68 <a href="#">__gnu_cxx::stdio_sync_filebuf&lt;_CharT, _Traits&gt; Class Template Reference</a>	1071
5.68.1 Detailed Description	1074
5.68.2 Member Function Documentation	1074
5.68.3 Member Data Documentation	1091
5.69 <a href="#">__gnu_cxx::subtractive_rng Class Reference</a>	1093
5.69.1 Detailed Description	1094
5.69.2 Member Typedef Documentation	1094

5.69.3 Constructor & Destructor Documentation	1094
5.69.4 Member Function Documentation	1095
5.70 <code>__gnu_cxx::temporary_buffer&lt;_ForwardIterator, _Tp&gt;</code> Struct Template Reference	1095
5.70.1 Detailed Description	1096
5.70.2 Constructor & Destructor Documentation	1096
5.70.3 Member Function Documentation	1097
5.71 <code>__gnu_cxx::throw_allocator_base&lt;_Tp, _Cond&gt;</code> Class Template Reference	1098
5.71.1 Detailed Description	1099
5.72 <code>__gnu_cxx::throw_allocator_limit&lt;_Tp&gt;</code> Struct Template Reference	1100
5.72.1 Detailed Description	1101
5.73 <code>__gnu_cxx::throw_allocator_random&lt;_Tp&gt;</code> Struct Template Reference	1102
5.73.1 Detailed Description	1103
5.74 <code>__gnu_cxx::throw_value_base&lt;_Cond&gt;</code> Struct Template Reference	1103
5.74.1 Detailed Description	1104
5.75 <code>__gnu_cxx::throw_value_limit</code> Struct Reference	1105
5.75.1 Detailed Description	1106
5.76 <code>__gnu_cxx::throw_value_random</code> Struct Reference	1106
5.76.1 Detailed Description	1107
5.77 <code>__gnu_cxx::unary_compose&lt;_Operation1, _Operation2&gt;</code> Class Template Reference	1107
5.77.1 Detailed Description	1108
5.77.2 Member Typedef Documentation	1108
5.78 <code>__gnu_debug::_After_nth_from&lt;_Iterator&gt;</code> Class Template Reference	1109
5.78.1 Detailed Description	1109
5.79 <code>__gnu_debug::_BeforeBeginHelper&lt;_Sequence&gt;</code> Struct Template Reference	1109
5.79.1 Detailed Description	1109
5.80 <code>__gnu_debug::_Equal_to&lt;_Type&gt;</code> Class Template Reference	1110
5.80.1 Detailed Description	1110
5.81 <code>__gnu_debug::_Not_equal_to&lt;_Type&gt;</code> Class Template Reference	1110
5.81.1 Detailed Description	1110
5.82 <code>__gnu_debug::_Safe_container&lt;_SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware&gt;</code> Class Template Reference	1110
5.82.1 Detailed Description	1111
5.83 <code>__gnu_debug::_Safe_forward_list&lt;_SafeSequence&gt;</code> Class Template Reference	1111
5.83.1 Detailed Description	1112
5.83.2 Member Function Documentation	1112
5.83.3 Member Data Documentation	1114
5.84 <code>__gnu_debug::_Safe_iterator&lt;_Iterator, _Sequence&gt;</code> Class Template Reference	1115
5.84.1 Detailed Description	1117
5.84.2 Constructor & Destructor Documentation	1117

5.84.3 Member Function Documentation . . . . .	1119
5.84.4 Member Data Documentation . . . . .	1126
5.85 <a href="#">__gnu_debug::_Safe_iterator_base Class Reference</a> . . . . .	1128
5.85.1 Detailed Description . . . . .	1129
5.85.2 Constructor & Destructor Documentation . . . . .	1129
5.85.3 Member Function Documentation . . . . .	1130
5.85.4 Member Data Documentation . . . . .	1132
5.86 <a href="#">__gnu_debug::_Safe_local_iterator&lt; _Iterator, _Sequence &gt; Class Template Reference</a> . . . . .	1133
5.86.1 Detailed Description . . . . .	1135
5.86.2 Constructor & Destructor Documentation . . . . .	1135
5.86.3 Member Function Documentation . . . . .	1137
5.86.4 Member Data Documentation . . . . .	1144
5.87 <a href="#">__gnu_debug::_Safe_local_iterator_base Class Reference</a> . . . . .	1145
5.87.1 Detailed Description . . . . .	1146
5.87.2 Constructor & Destructor Documentation . . . . .	1146
5.87.3 Member Function Documentation . . . . .	1147
5.87.4 Member Data Documentation . . . . .	1149
5.88 <a href="#">__gnu_debug::_Safe_node_sequence&lt; _Sequence &gt; Class Template Reference</a> . . . . .	1150
5.88.1 Detailed Description . . . . .	1151
5.88.2 Member Function Documentation . . . . .	1151
5.88.3 Member Data Documentation . . . . .	1153
5.89 <a href="#">__gnu_debug::_Safe_sequence&lt; _Sequence &gt; Class Template Reference</a> . . . . .	1154
5.89.1 Detailed Description . . . . .	1155
5.89.2 Member Function Documentation . . . . .	1155
5.89.3 Member Data Documentation . . . . .	1157
5.90 <a href="#">__gnu_debug::_Safe_sequence_base Class Reference</a> . . . . .	1158
5.90.1 Detailed Description . . . . .	1159
5.90.2 Constructor & Destructor Documentation . . . . .	1159
5.90.3 Member Function Documentation . . . . .	1159
5.90.4 Member Data Documentation . . . . .	1161
5.91 <a href="#">__gnu_debug::_Safe_unordered_container&lt; _Container &gt; Class Template Reference</a> . . . . .	1162
5.91.1 Detailed Description . . . . .	1163
5.91.2 Member Function Documentation . . . . .	1163
5.91.3 Member Data Documentation . . . . .	1165
5.92 <a href="#">__gnu_debug::_Safe_unordered_container_base Class Reference</a> . . . . .	1166
5.92.1 Detailed Description . . . . .	1167
5.92.2 Constructor & Destructor Documentation . . . . .	1167
5.92.3 Member Function Documentation . . . . .	1167
5.92.4 Member Data Documentation . . . . .	1169

5.93	<a href="#">__gnu_debug::__Safe_vector&lt;_SafeSequence, _BaseSequence &gt; Class Template Reference</a>	1170
5.93.1	Detailed Description	1170
5.94	<a href="#">__gnu_debug::__Sequence_traits&lt;_Sequence &gt; Struct Template Reference</a>	1171
5.94.1	Detailed Description	1171
5.95	<a href="#">__gnu_debug::basic_string&lt;_CharT, _Traits, _Allocator &gt; Class Template Reference</a>	1171
5.95.1	Detailed Description	1176
5.95.2	Member Function Documentation	1176
5.95.3	Member Data Documentation	1200
5.96	<a href="#">__gnu_parallel::__accumulate_binop_reduct&lt;_BinOp &gt; Struct Template Reference</a>	1201
5.96.1	Detailed Description	1201
5.97	<a href="#">__gnu_parallel::__accumulate_selector&lt;_It &gt; Struct Template Reference</a>	1202
5.97.1	Detailed Description	1202
5.97.2	Member Function Documentation	1202
5.97.3	Member Data Documentation	1203
5.98	<a href="#">__gnu_parallel::__adjacent_difference_selector&lt;_It &gt; Struct Template Reference</a>	1203
5.98.1	Detailed Description	1204
5.98.2	Member Data Documentation	1204
5.99	<a href="#">__gnu_parallel::__adjacent_find_selector Struct Reference</a>	1204
5.99.1	Detailed Description	1205
5.99.2	Member Function Documentation	1205
5.100	<a href="#">__gnu_parallel::__binder1st&lt;_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType &gt; Class Template Reference</a>	1206
5.100.1	Detailed Description	1207
5.100.2	Member Typedef Documentation	1207
5.101	<a href="#">__gnu_parallel::__binder2nd&lt;_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType &gt; Class Template Reference</a>	1208
5.101.1	Detailed Description	1208
5.101.2	Member Typedef Documentation	1209
5.102	<a href="#">__gnu_parallel::__count_if_selector&lt;_It, _Diff &gt; Struct Template Reference</a>	1209
5.102.1	Detailed Description	1210
5.102.2	Member Function Documentation	1210
5.102.3	Member Data Documentation	1211
5.103	<a href="#">__gnu_parallel::__count_selector&lt;_It, _Diff &gt; Struct Template Reference</a>	1211
5.103.1	Detailed Description	1212
5.103.2	Member Function Documentation	1212
5.103.3	Member Data Documentation	1212
5.104	<a href="#">__gnu_parallel::__fill_selector&lt;_It &gt; Struct Template Reference</a>	1213
5.104.1	Detailed Description	1213
5.104.2	Member Function Documentation	1213
5.104.3	Member Data Documentation	1214

5.105 <a href="#">__gnu_parallel::__find_first_of_selector&lt;_FIterator&gt; Struct Template Reference</a>	1214
5.105.1 Detailed Description	1215
5.105.2 Member Function Documentation	1215
5.106 <a href="#">__gnu_parallel::__find_if_selector Struct Reference</a>	1216
5.106.1 Detailed Description	1217
5.106.2 Member Function Documentation	1217
5.107 <a href="#">__gnu_parallel::__for_each_selector&lt;_It&gt; Struct Template Reference</a>	1218
5.107.1 Detailed Description	1219
5.107.2 Member Function Documentation	1219
5.107.3 Member Data Documentation	1219
5.108 <a href="#">__gnu_parallel::__generate_selector&lt;_It&gt; Struct Template Reference</a>	1220
5.108.1 Detailed Description	1220
5.108.2 Member Function Documentation	1220
5.108.3 Member Data Documentation	1221
5.109 <a href="#">__gnu_parallel::__generic_find_selector Struct Reference</a>	1221
5.109.1 Detailed Description	1222
5.110 <a href="#">__gnu_parallel::__generic_for_each_selector&lt;_It&gt; Struct Template Reference</a>	1223
5.110.1 Detailed Description	1224
5.110.2 Member Data Documentation	1224
5.111 <a href="#">__gnu_parallel::__identity_selector&lt;_It&gt; Struct Template Reference</a>	1224
5.111.1 Detailed Description	1225
5.111.2 Member Function Documentation	1225
5.111.3 Member Data Documentation	1226
5.112 <a href="#">__gnu_parallel::__inner_product_selector&lt;_It, _It2, _Tp&gt; Struct Template Reference</a>	1226
5.112.1 Detailed Description	1227
5.112.2 Constructor & Destructor Documentation	1227
5.112.3 Member Function Documentation	1227
5.112.4 Member Data Documentation	1228
5.113 <a href="#">__gnu_parallel::__max_element_reduct&lt;_Compare, _It&gt; Struct Template Reference</a>	1229
5.113.1 Detailed Description	1229
5.114 <a href="#">__gnu_parallel::__min_element_reduct&lt;_Compare, _It&gt; Struct Template Reference</a>	1229
5.114.1 Detailed Description	1229
5.115 <a href="#">__gnu_parallel::__mismatch_selector Struct Reference</a>	1230
5.115.1 Detailed Description	1230
5.115.2 Member Function Documentation	1230
5.116 <a href="#">__gnu_parallel::__multiway_merge_3_variant_sentinel_switch&lt;__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt; Struct Template Reference</a>	1231
5.116.1 Detailed Description	1232
5.117 <a href="#">__gnu_parallel::__multiway_merge_3_variant_sentinel_switch&lt;true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare&gt; Struct Template Reference</a>	1232

5.117.1 Detailed Description	1232
5.118 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt; __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt;</code> Struct Template Reference	1232
5.118.1 Detailed Description	1233
5.119 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch&lt; true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt;</code> Struct Template Reference	1233
5.119.1 Detailed Description	1233
5.120 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt; __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt;</code> Struct Template Reference	1233
5.120.1 Detailed Description	1234
5.121 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch&lt; false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare &gt;</code> Struct Template Reference	1234
5.121.1 Detailed Description	1234
5.122 <code>__gnu_parallel::__replace_if_selector&lt; _It, _Op, _Tp &gt;</code> Struct Template Reference	1235
5.122.1 Detailed Description	1235
5.122.2 Constructor & Destructor Documentation	1235
5.122.3 Member Function Documentation	1236
5.122.4 Member Data Documentation	1236
5.123 <code>__gnu_parallel::__replace_selector&lt; _It, _Tp &gt;</code> Struct Template Reference	1237
5.123.1 Detailed Description	1237
5.123.2 Constructor & Destructor Documentation	1238
5.123.3 Member Function Documentation	1238
5.123.4 Member Data Documentation	1238
5.124 <code>__gnu_parallel::__transform1_selector&lt; _It &gt;</code> Struct Template Reference	1239
5.124.1 Detailed Description	1240
5.124.2 Member Function Documentation	1240
5.124.3 Member Data Documentation	1240
5.125 <code>__gnu_parallel::__transform2_selector&lt; _It &gt;</code> Struct Template Reference	1241
5.125.1 Detailed Description	1241
5.125.2 Member Function Documentation	1242
5.125.3 Member Data Documentation	1242
5.126 <code>__gnu_parallel::__unary_negate&lt; _Predicate, argument_type &gt;</code> Class Template Reference	1243
5.126.1 Detailed Description	1243
5.126.2 Member Typedef Documentation	1243
5.127 <code>__gnu_parallel::_DRandomShufflingGlobalData&lt; _RAIter &gt;</code> Struct Template Reference	1244
5.127.1 Detailed Description	1245
5.127.2 Constructor & Destructor Documentation	1245
5.127.3 Member Data Documentation	1245
5.128 <code>__gnu_parallel::_DRSSorterPU&lt; _RAIter, _RandomNumberGenerator &gt;</code> Struct Template Reference	1247
5.128.1 Detailed Description	1247

5.128.2 Member Data Documentation	1247
5.129 <code>__gnu_parallel::_DummyReduct</code> Struct Reference	1249
5.129.1 Detailed Description	1249
5.130 <code>__gnu_parallel::_EqualFromLess&lt;_T1, _T2, _Compare&gt;</code> Class Template Reference	1249
5.130.1 Detailed Description	1250
5.130.2 Member Typedef Documentation	1250
5.131 <code>__gnu_parallel::_EqualTo&lt;_T1, _T2&gt;</code> Struct Template Reference	1251
5.131.1 Detailed Description	1251
5.131.2 Member Typedef Documentation	1251
5.132 <code>__gnu_parallel::_GuardedIterator&lt;_RAIter, _Compare&gt;</code> Class Template Reference	1252
5.132.1 Detailed Description	1253
5.132.2 Constructor & Destructor Documentation	1253
5.132.3 Member Function Documentation	1253
5.132.4 Friends And Related Function Documentation	1254
5.133 <code>__gnu_parallel::_IteratorPair&lt;_Iterator1, _Iterator2, _IteratorCategory&gt;</code> Class Template Reference	1256
5.133.1 Detailed Description	1257
5.133.2 Member Typedef Documentation	1257
5.133.3 Member Data Documentation	1258
5.134 <code>__gnu_parallel::_IteratorTriple&lt;_Iterator1, _Iterator2, _Iterator3, _IteratorCategory&gt;</code> Class Template Reference	1258
5.134.1 Detailed Description	1259
5.135 <code>__gnu_parallel::_Job&lt;_DifferenceTp&gt;</code> Struct Template Reference	1259
5.135.1 Detailed Description	1259
5.135.2 Member Data Documentation	1259
5.136 <code>__gnu_parallel::_Less&lt;_T1, _T2&gt;</code> Struct Template Reference	1261
5.136.1 Detailed Description	1261
5.136.2 Member Typedef Documentation	1261
5.137 <code>__gnu_parallel::_Lexicographic&lt;_T1, _T2, _Compare&gt;</code> Class Template Reference	1262
5.137.1 Detailed Description	1263
5.137.2 Member Typedef Documentation	1263
5.138 <code>__gnu_parallel::_LexicographicReverse&lt;_T1, _T2, _Compare&gt;</code> Class Template Reference	1264
5.138.1 Detailed Description	1265
5.138.2 Member Typedef Documentation	1265
5.139 <code>__gnu_parallel::_LoserTree&lt;__stable, _Tp, _Compare&gt;</code> Class Template Reference	1266
5.139.1 Detailed Description	1266
5.139.2 Member Function Documentation	1267
5.139.3 Member Data Documentation	1268
5.140 <code>__gnu_parallel::_LoserTree&lt;false, _Tp, _Compare&gt;</code> Class Template Reference	1268
5.140.1 Detailed Description	1269

5.140.2 Member Function Documentation	1269
5.141 <code>__gnu_parallel::LoserTreeBase&lt; _Tp, _Compare &gt;</code> Class Template Reference	1271
5.141.1 Detailed Description	1272
5.141.2 Constructor & Destructor Documentation	1272
5.141.3 Member Function Documentation	1273
5.141.4 Member Data Documentation	1274
5.142 <code>__gnu_parallel::LoserTreeBase&lt; _Tp, _Compare &gt;::_Loser</code> Struct Reference	1275
5.142.1 Detailed Description	1275
5.142.2 Member Data Documentation	1275
5.143 <code>__gnu_parallel::LoserTreePointer&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference	1276
5.143.1 Detailed Description	1277
5.144 <code>__gnu_parallel::LoserTreePointer&lt; false, _Tp, _Compare &gt;</code> Class Template Reference	1277
5.144.1 Detailed Description	1278
5.145 <code>__gnu_parallel::LoserTreePointerBase&lt; _Tp, _Compare &gt;</code> Class Template Reference	1278
5.145.1 Detailed Description	1279
5.146 <code>__gnu_parallel::LoserTreePointerBase&lt; _Tp, _Compare &gt;::_Loser</code> Struct Reference	1279
5.146.1 Detailed Description	1280
5.147 <code>__gnu_parallel::LoserTreePointerUnguarded&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference	1280
5.147.1 Detailed Description	1281
5.148 <code>__gnu_parallel::LoserTreePointerUnguarded&lt; false, _Tp, _Compare &gt;</code> Class Template Reference	1281
5.148.1 Detailed Description	1282
5.149 <code>__gnu_parallel::LoserTreePointerUnguardedBase&lt; _Tp, _Compare &gt;</code> Class Template Reference	1282
5.149.1 Detailed Description	1283
5.150 <code>__gnu_parallel::LoserTreeTraits&lt; _Tp &gt;</code> Struct Template Reference	1283
5.150.1 Detailed Description	1283
5.150.2 Member Data Documentation	1284
5.151 <code>__gnu_parallel::LoserTreeUnguarded&lt; __stable, _Tp, _Compare &gt;</code> Class Template Reference	1284
5.151.1 Detailed Description	1285
5.152 <code>__gnu_parallel::LoserTreeUnguarded&lt; false, _Tp, _Compare &gt;</code> Class Template Reference	1285
5.152.1 Detailed Description	1286
5.153 <code>__gnu_parallel::LoserTreeUnguardedBase&lt; _Tp, _Compare &gt;</code> Class Template Reference	1286
5.153.1 Detailed Description	1287
5.154 <code>__gnu_parallel::Multiplies&lt; _Tp1, _Tp2, _Result &gt;</code> Struct Template Reference	1287
5.154.1 Detailed Description	1288
5.154.2 Member Typedef Documentation	1288
5.155 <code>__gnu_parallel::Nothing</code> Struct Reference	1289
5.155.1 Detailed Description	1289
5.155.2 Member Function Documentation	1289
5.156 <code>__gnu_parallel::Piece&lt; _DifferenceTp &gt;</code> Struct Template Reference	1290



5.156.1 Detailed Description	1290
5.156.2 Member Data Documentation	1290
5.157 <code>__gnu_parallel::Plus&lt; _Tp1, _Tp2, _Result &gt;</code> Struct Template Reference	1291
5.157.1 Detailed Description	1291
5.157.2 Member Typedef Documentation	1292
5.158 <code>__gnu_parallel::PMWMSSortingData&lt; _RAIter &gt;</code> Struct Template Reference	1292
5.158.1 Detailed Description	1293
5.158.2 Member Data Documentation	1293
5.159 <code>__gnu_parallel::PseudoSequence&lt; _Tp, _DifferenceTp &gt;</code> Class Template Reference	1295
5.159.1 Detailed Description	1295
5.159.2 Constructor & Destructor Documentation	1296
5.159.3 Member Function Documentation	1296
5.160 <code>__gnu_parallel::PseudoSequenceIterator&lt; _Tp, _DifferenceTp &gt;</code> Class Template Reference	1297
5.160.1 Detailed Description	1297
5.161 <code>__gnu_parallel::QSBThreadLocal&lt; _RAIter &gt;</code> Struct Template Reference	1297
5.161.1 Detailed Description	1298
5.161.2 Member Typedef Documentation	1298
5.161.3 Constructor & Destructor Documentation	1298
5.161.4 Member Data Documentation	1299
5.162 <code>__gnu_parallel::RandomNumber</code> Class Reference	1300
5.162.1 Detailed Description	1300
5.162.2 Constructor & Destructor Documentation	1300
5.162.3 Member Function Documentation	1301
5.163 <code>__gnu_parallel::RestrictedBoundedConcurrentQueue&lt; _Tp &gt;</code> Class Template Reference	1302
5.163.1 Detailed Description	1302
5.163.2 Constructor & Destructor Documentation	1303
5.163.3 Member Function Documentation	1303
5.164 <code>__gnu_parallel::SamplingSorter&lt; __stable, _RAIter, _StrictWeakOrdering &gt;</code> Struct Template Reference	1304
5.164.1 Detailed Description	1304
5.165 <code>__gnu_parallel::SamplingSorter&lt; false, _RAIter, _StrictWeakOrdering &gt;</code> Struct Template Reference	1305
5.165.1 Detailed Description	1305
5.166 <code>__gnu_parallel::Settings</code> Struct Reference	1305
5.166.1 Detailed Description	1306
5.166.2 Member Function Documentation	1306
5.166.3 Member Data Documentation	1307
5.167 <code>__gnu_parallel::SplitConsistently&lt; __exact, _RAIter, _Compare, _SortingPlacesIterator &gt;</code> Struct Template Reference	1316
5.167.1 Detailed Description	1316
5.168 <code>__gnu_parallel::SplitConsistently&lt; false, _RAIter, _Compare, _SortingPlacesIterator &gt;</code> Struct Template Reference	1316

5.168.1 Detailed Description	1317
5.169 <code>__gnu_parallel::SplitConsistently&lt; true, _RAIter, _Compare, _SortingPlacesIterator &gt;</code> Struct Template Reference	1317
5.169.1 Detailed Description	1317
5.170 <code>__gnu_parallel::balanced_quicksort_tag</code> Struct Reference	1317
5.170.1 Detailed Description	1318
5.170.2 Member Function Documentation	1318
5.171 <code>__gnu_parallel::balanced_tag</code> Struct Reference	1319
5.171.1 Detailed Description	1319
5.171.2 Member Function Documentation	1319
5.172 <code>__gnu_parallel::constant_size_blocks_tag</code> Struct Reference	1320
5.172.1 Detailed Description	1320
5.173 <code>__gnu_parallel::default_parallel_tag</code> Struct Reference	1321
5.173.1 Detailed Description	1321
5.173.2 Member Function Documentation	1321
5.174 <code>__gnu_parallel::equal_split_tag</code> Struct Reference	1322
5.174.1 Detailed Description	1322
5.175 <code>__gnu_parallel::exact_tag</code> Struct Reference	1323
5.175.1 Detailed Description	1323
5.175.2 Member Function Documentation	1323
5.176 <code>__gnu_parallel::find_tag</code> Struct Reference	1324
5.176.1 Detailed Description	1324
5.177 <code>__gnu_parallel::growing_blocks_tag</code> Struct Reference	1325
5.177.1 Detailed Description	1325
5.178 <code>__gnu_parallel::multiway_mergesort_exact_tag</code> Struct Reference	1325
5.178.1 Detailed Description	1326
5.178.2 Member Function Documentation	1326
5.179 <code>__gnu_parallel::multiway_mergesort_sampling_tag</code> Struct Reference	1327
5.179.1 Detailed Description	1327
5.179.2 Member Function Documentation	1327
5.180 <code>__gnu_parallel::multiway_mergesort_tag</code> Struct Reference	1328
5.180.1 Detailed Description	1328
5.180.2 Member Function Documentation	1329
5.181 <code>__gnu_parallel::omp_loop_static_tag</code> Struct Reference	1330
5.181.1 Detailed Description	1330
5.181.2 Member Function Documentation	1330
5.182 <code>__gnu_parallel::omp_loop_tag</code> Struct Reference	1331
5.182.1 Detailed Description	1331
5.182.2 Member Function Documentation	1331

5.183	<a href="#">__gnu_parallel::parallel_tag Struct Reference</a>	1333
5.183.1	Detailed Description	1334
5.183.2	Constructor & Destructor Documentation	1334
5.183.3	Member Function Documentation	1334
5.184	<a href="#">__gnu_parallel::quicksort_tag Struct Reference</a>	1335
5.184.1	Detailed Description	1335
5.184.2	Member Function Documentation	1336
5.185	<a href="#">__gnu_parallel::sampling_tag Struct Reference</a>	1337
5.185.1	Detailed Description	1337
5.185.2	Member Function Documentation	1337
5.186	<a href="#">__gnu_parallel::sequential_tag Struct Reference</a>	1338
5.186.1	Detailed Description	1338
5.187	<a href="#">__gnu_parallel::unbalanced_tag Struct Reference</a>	1338
5.187.1	Detailed Description	1339
5.187.2	Member Function Documentation	1339
5.188	<a href="#">__gnu_pbds::associative_tag Struct Reference</a>	1340
5.188.1	Detailed Description	1340
5.189	<a href="#">__gnu_pbds::basic_branch&lt; Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc &gt; Class Template Reference</a>	1340
5.189.1	Detailed Description	1341
5.190	<a href="#">__gnu_pbds::basic_branch_tag Struct Reference</a>	1341
5.190.1	Detailed Description	1341
5.191	<a href="#">__gnu_pbds::basic_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc &gt; Class Template Reference</a>	1342
5.191.1	Detailed Description	1342
5.192	<a href="#">__gnu_pbds::basic_hash_tag Struct Reference</a>	1343
5.192.1	Detailed Description	1343
5.193	<a href="#">__gnu_pbds::basic_invalidation_guarantee Struct Reference</a>	1344
5.193.1	Detailed Description	1344
5.194	<a href="#">__gnu_pbds::binary_heap_tag Struct Reference</a>	1345
5.194.1	Detailed Description	1345
5.195	<a href="#">__gnu_pbds::binomial_heap_tag Struct Reference</a>	1346
5.195.1	Detailed Description	1346
5.196	<a href="#">__gnu_pbds::cc_hash_max_collision_check_resize_trigger&lt; External_Load_Access, Size_Type &gt; Class Template Reference</a>	1346
5.196.1	Detailed Description	1347
5.196.2	Member Enumeration Documentation	1347
5.196.3	Constructor & Destructor Documentation	1348
5.196.4	Member Function Documentation	1348

5.197	<a href="#">__gnu_pbds::cc_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc &gt; Class Template Reference</a>	1353
5.197.1	Detailed Description	1354
5.197.2	Constructor & Destructor Documentation	1354
5.198	<a href="#">__gnu_pbds::cc_hash_tag Struct Reference</a>	1358
5.198.1	Detailed Description	1359
5.199	<a href="#">__gnu_pbds::container_error Struct Reference</a>	1359
5.199.1	Detailed Description	1359
5.199.2	Member Function Documentation	1360
5.200	<a href="#">__gnu_pbds::container_tag Struct Reference</a>	1360
5.200.1	Detailed Description	1360
5.201	<a href="#">__gnu_pbds::container_traits&lt; Cntnr &gt; Struct Template Reference</a>	1361
5.201.1	Detailed Description	1361
5.201.2	Member Enumeration Documentation	1361
5.202	<a href="#">__gnu_pbds::container_traits_base&lt; _Tag &gt; Struct Template Reference</a>	1362
5.202.1	Detailed Description	1362
5.203	<a href="#">__gnu_pbds::container_traits_base&lt; binary_heap_tag &gt; Struct Template Reference</a>	1362
5.203.1	Detailed Description	1362
5.204	<a href="#">__gnu_pbds::container_traits_base&lt; binomial_heap_tag &gt; Struct Template Reference</a>	1363
5.204.1	Detailed Description	1363
5.205	<a href="#">__gnu_pbds::container_traits_base&lt; cc_hash_tag &gt; Struct Template Reference</a>	1363
5.205.1	Detailed Description	1363
5.206	<a href="#">__gnu_pbds::container_traits_base&lt; gp_hash_tag &gt; Struct Template Reference</a>	1364
5.206.1	Detailed Description	1364
5.207	<a href="#">__gnu_pbds::container_traits_base&lt; list_update_tag &gt; Struct Template Reference</a>	1364
5.207.1	Detailed Description	1364
5.208	<a href="#">__gnu_pbds::container_traits_base&lt; ov_tree_tag &gt; Struct Template Reference</a>	1365
5.208.1	Detailed Description	1365
5.209	<a href="#">__gnu_pbds::container_traits_base&lt; pairing_heap_tag &gt; Struct Template Reference</a>	1365
5.209.1	Detailed Description	1365
5.210	<a href="#">__gnu_pbds::container_traits_base&lt; pat_trie_tag &gt; Struct Template Reference</a>	1366
5.210.1	Detailed Description	1366
5.211	<a href="#">__gnu_pbds::container_traits_base&lt; rb_tree_tag &gt; Struct Template Reference</a>	1366
5.211.1	Detailed Description	1366
5.212	<a href="#">__gnu_pbds::container_traits_base&lt; rc_binomial_heap_tag &gt; Struct Template Reference</a>	1367
5.212.1	Detailed Description	1367
5.213	<a href="#">__gnu_pbds::container_traits_base&lt; splay_tree_tag &gt; Struct Template Reference</a>	1367
5.213.1	Detailed Description	1367
5.214	<a href="#">__gnu_pbds::container_traits_base&lt; thin_heap_tag &gt; Struct Template Reference</a>	1368

5.214.1 Detailed Description	1368
5.215 <a href="#">__gnu_pbds::detail::bin_search_tree_const_it_&lt; Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc &gt; Class Template Reference</a>	1368
5.215.1 Detailed Description	1369
5.216 <a href="#">__gnu_pbds::detail::bin_search_tree_const_node_it_&lt; Node, Const_Iterator, Iterator, _Alloc &gt; Class Template Reference</a>	1370
5.216.1 Detailed Description	1370
5.216.2 Member Typedef Documentation	1371
5.216.3 Member Function Documentation	1372
5.217 <a href="#">__gnu_pbds::detail::bin_search_tree_it_&lt; Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc &gt; Class Template Reference</a>	1374
5.217.1 Detailed Description	1376
5.218 <a href="#">__gnu_pbds::detail::bin_search_tree_node_it_&lt; Node, Const_Iterator, Iterator, _Alloc &gt; Class Template Reference</a>	1376
5.218.1 Detailed Description	1377
5.218.2 Member Typedef Documentation	1377
5.218.3 Member Function Documentation	1378
5.219 <a href="#">__gnu_pbds::detail::bin_search_tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc &gt; Struct Template Reference</a>	1380
5.219.1 Detailed Description	1381
5.219.2 Member Typedef Documentation	1381
5.220 <a href="#">__gnu_pbds::detail::bin_search_tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc &gt; Struct Template Reference</a>	1381
5.220.1 Detailed Description	1382
5.220.2 Member Typedef Documentation	1382
5.221 <a href="#">__gnu_pbds::detail::binary_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1382
5.221.1 Detailed Description	1384
5.222 <a href="#">__gnu_pbds::detail::binary_heap_const_iterator_&lt; Value_Type, Entry, Simple, _Alloc &gt; Class Template Reference</a>	1384
5.222.1 Detailed Description	1385
5.222.2 Member Typedef Documentation	1385
5.222.3 Constructor & Destructor Documentation	1387
5.222.4 Member Function Documentation	1388
5.223 <a href="#">__gnu_pbds::detail::binary_heap_point_const_iterator_&lt; Value_Type, Entry, Simple, _Alloc &gt; Class Template Reference</a>	1390
5.223.1 Detailed Description	1391
5.223.2 Member Typedef Documentation	1391
5.223.3 Constructor & Destructor Documentation	1392
5.223.4 Member Function Documentation	1393
5.224 <a href="#">__gnu_pbds::detail::binomial_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1394
5.224.1 Detailed Description	1396

5.225	<a href="#">__gnu_pbds::detail::binomial_heap_base&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1396
5.225.1	<a href="#">Detailed Description</a>	1398
5.226	<a href="#">__gnu_pbds::detail::branch_policy&lt; Node_Cltr, Node_Itr, _Alloc &gt; Struct Template Reference</a>	1399
5.226.1	<a href="#">Detailed Description</a>	1399
5.227	<a href="#">__gnu_pbds::detail::branch_policy&lt; Node_Cltr, Node_Cltr, _Alloc &gt; Struct Template Reference</a>	1400
5.227.1	<a href="#">Detailed Description</a>	1400
5.228	<a href="#">__gnu_pbds::detail::cc_ht_map&lt; Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy &gt; Class Template Reference</a>	1401
5.228.1	<a href="#">Detailed Description</a>	1403
5.228.2	<a href="#">Member Enumeration Documentation</a>	1403
5.228.3	<a href="#">Member Function Documentation</a>	1403
5.229	<a href="#">__gnu_pbds::detail::cond_dealtor&lt; Entry, _Alloc &gt; Class Template Reference</a>	1406
5.229.1	<a href="#">Detailed Description</a>	1406
5.230	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, Tag, Policy_TI &gt; Struct Template Reference</a>	1407
5.230.1	<a href="#">Detailed Description</a>	1407
5.231	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type &gt; Struct Template Reference</a>	1407
5.231.1	<a href="#">Detailed Description</a>	1407
5.231.2	<a href="#">Member Typedef Documentation</a>	1407
5.232	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type &gt; Struct Template Reference</a>	1408
5.232.1	<a href="#">Detailed Description</a>	1408
5.232.2	<a href="#">Member Typedef Documentation</a>	1408
5.233	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type &gt; Struct Template Reference</a>	1408
5.233.1	<a href="#">Detailed Description</a>	1409
5.233.2	<a href="#">Member Typedef Documentation</a>	1409
5.234	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type &gt; Struct Template Reference</a>	1409
5.234.1	<a href="#">Detailed Description</a>	1409
5.234.2	<a href="#">Member Typedef Documentation</a>	1410
5.235	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type &gt; Struct Template Reference</a>	1410
5.235.1	<a href="#">Detailed Description</a>	1410
5.235.2	<a href="#">Member Typedef Documentation</a>	1410
5.236	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, cc_hash_tag, Policy_TI &gt; Struct Template Reference</a>	1411
5.236.1	<a href="#">Detailed Description</a>	1411
5.236.2	<a href="#">Member Typedef Documentation</a>	1411
5.237	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, gp_hash_tag, Policy_TI &gt; Struct Template Reference</a>	1411

5.237.1 Detailed Description . . . . .	1412
5.237.2 Member Typedef Documentation . . . . .	1412
5.238 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, list_update_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1412
5.238.1 Detailed Description . . . . .	1412
5.238.2 Member Typedef Documentation . . . . .	1413
5.239 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, ov_tree_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1413
5.239.1 Detailed Description . . . . .	1413
5.239.2 Member Typedef Documentation . . . . .	1413
5.240 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, pat_trie_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1414
5.240.1 Detailed Description . . . . .	1414
5.241 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, rb_tree_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1414
5.241.1 Detailed Description . . . . .	1414
5.241.2 Member Typedef Documentation . . . . .	1414
5.242 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, Mapped, _Alloc, splay_tree_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1415
5.242.1 Detailed Description . . . . .	1415
5.242.2 Member Typedef Documentation . . . . .	1415
5.243 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, cc_hash_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1416
5.243.1 Detailed Description . . . . .	1416
5.243.2 Member Typedef Documentation . . . . .	1416
5.244 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, gp_hash_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1416
5.244.1 Detailed Description . . . . .	1417
5.244.2 Member Typedef Documentation . . . . .	1417
5.245 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, list_update_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1417
5.245.1 Detailed Description . . . . .	1417
5.245.2 Member Typedef Documentation . . . . .	1418
5.246 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, ov_tree_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1418
5.246.1 Detailed Description . . . . .	1418
5.246.2 Member Typedef Documentation . . . . .	1418
5.247 <code>__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, pat_trie_tag, Policy_TI &gt;</code> Struct Template Reference . . . . .	1419
5.247.1 Detailed Description . . . . .	1419
5.247.2 Member Typedef Documentation . . . . .	1419

5.248	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, rb_tree_tag, Policy_Tl &gt; Struct Template Reference</a>	1419
5.248.1	Detailed Description	1420
5.249	<a href="#">__gnu_pbds::detail::container_base_dispatch&lt; Key, null_type, _Alloc, splay_tree_tag, Policy_Tl &gt; Struct Template Reference</a>	1420
5.249.1	Detailed Description	1420
5.249.2	Member Typedef Documentation	1420
5.250	<a href="#">__gnu_pbds::detail::default_comb_hash_fn Struct Reference</a>	1421
5.250.1	Detailed Description	1421
5.250.2	Member Typedef Documentation	1421
5.251	<a href="#">__gnu_pbds::detail::default_eq_fn&lt; Key &gt; Struct Template Reference</a>	1421
5.251.1	Detailed Description	1421
5.251.2	Member Typedef Documentation	1422
5.252	<a href="#">__gnu_pbds::detail::default_hash_fn&lt; Key &gt; Struct Template Reference</a>	1422
5.252.1	Detailed Description	1422
5.252.2	Member Typedef Documentation	1422
5.253	<a href="#">__gnu_pbds::detail::default_probe_fn&lt; Comb_Probe_Fn &gt; Struct Template Reference</a>	1423
5.253.1	Detailed Description	1423
5.253.2	Member Typedef Documentation	1423
5.254	<a href="#">__gnu_pbds::detail::default_resize_policy&lt; Comb_Hash_Fn &gt; Struct Template Reference</a>	1423
5.254.1	Detailed Description	1423
5.254.2	Member Typedef Documentation	1424
5.255	<a href="#">__gnu_pbds::detail::default_trie_access_traits&lt; Key &gt; Struct Template Reference</a>	1424
5.255.1	Detailed Description	1424
5.256	<a href="#">__gnu_pbds::detail::default_trie_access_traits&lt; std::basic_string&lt; Char, Char_Traits, std::allocator&lt; char &gt; &gt; &gt; Struct Template Reference</a>	1424
5.256.1	Detailed Description	1424
5.256.2	Member Typedef Documentation	1425
5.257	<a href="#">__gnu_pbds::detail::default_update_policy Struct Reference</a>	1425
5.257.1	Detailed Description	1425
5.257.2	Member Typedef Documentation	1425
5.258	<a href="#">__gnu_pbds::detail::dumnode_const_iterator&lt; Key, Data, _Alloc &gt; Struct Template Reference</a>	1426
5.258.1	Detailed Description	1426
5.259	<a href="#">__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, No_Throw &gt; Struct Template Reference</a>	1426
5.259.1	Detailed Description	1426
5.260	<a href="#">__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, false &gt; Struct Template Reference</a>	1426
5.260.1	Detailed Description	1427
5.261	<a href="#">__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, false &gt;::type Struct Reference</a>	1427
5.261.1	Detailed Description	1427
5.262	<a href="#">__gnu_pbds::detail::entry_cmp&lt; _VTp, Cmp_Fn, _Alloc, true &gt; Struct Template Reference</a>	1427



5.262.1 Detailed Description . . . . .	1428
5.262.2 Member Typedef Documentation . . . . .	1428
5.263 <code>__gnu_pbds::detail::entry_pred&lt; _VTp, Pred, _Alloc, No_Throw &gt;</code> Struct Template Reference . . . . .	1428
5.263.1 Detailed Description . . . . .	1428
5.264 <code>__gnu_pbds::detail::entry_pred&lt; _VTp, Pred, _Alloc, false &gt;</code> Struct Template Reference . . . . .	1428
5.264.1 Detailed Description . . . . .	1429
5.265 <code>__gnu_pbds::detail::entry_pred&lt; _VTp, Pred, _Alloc, true &gt;</code> Struct Template Reference . . . . .	1429
5.265.1 Detailed Description . . . . .	1429
5.266 <code>__gnu_pbds::detail::eq_by_less&lt; Key, Cmp_Fn &gt;</code> Struct Template Reference . . . . .	1429
5.266.1 Detailed Description . . . . .	1430
5.267 <code>__gnu_pbds::detail::gp_ht_map&lt; Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe← _Fn, Probe_Fn, Resize_Policy &gt;</code> Class Template Reference . . . . .	1430
5.267.1 Detailed Description . . . . .	1432
5.267.2 Member Enumeration Documentation . . . . .	1433
5.267.3 Member Function Documentation . . . . .	1433
5.268 <code>__gnu_pbds::detail::hash_eq_fn&lt; Key, Eq_Fn, _Alloc, Store_Hash &gt;</code> Struct Template Reference . . . . .	1436
5.268.1 Detailed Description . . . . .	1436
5.269 <code>__gnu_pbds::detail::hash_eq_fn&lt; Key, Eq_Fn, _Alloc, false &gt;</code> Struct Template Reference . . . . .	1437
5.269.1 Detailed Description . . . . .	1437
5.270 <code>__gnu_pbds::detail::hash_eq_fn&lt; Key, Eq_Fn, _Alloc, true &gt;</code> Struct Template Reference . . . . .	1437
5.270.1 Detailed Description . . . . .	1438
5.271 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base&lt; Size_Type, Hold_Size &gt;</code> Class Template Reference . . . . .	1438
5.271.1 Detailed Description . . . . .	1438
5.272 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base&lt; Size_Type, true &gt;</code> Class Template Reference . . . . .	1438
5.272.1 Detailed Description . . . . .	1439
5.273 <code>__gnu_pbds::detail::left_child_next_sibling_heap&lt; Value_Type, Cmp_Fn, Node_Metadata, _Alloc &gt;</code> Class Template Reference . . . . .	1439
5.273.1 Detailed Description . . . . .	1440
5.274 <code>__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_&lt; Node, _Alloc &gt;</code> Class Template Reference . . . . .	1441
5.274.1 Detailed Description . . . . .	1442
5.274.2 Member Typedef Documentation . . . . .	1442
5.274.3 Constructor & Destructor Documentation . . . . .	1443
5.274.4 Member Function Documentation . . . . .	1444
5.275 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node_&lt; _Value, _Metadata, _Alloc &gt;</code> Struct Template Reference . . . . .	1446
5.275.1 Detailed Description . . . . .	1446
5.276 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_&lt; Node, _Alloc &gt;</code> Class Template Reference . . . . .	1447

5.276.1 Detailed Description	1448
5.276.2 Member Typedef Documentation	1448
5.276.3 Constructor & Destructor Documentation	1449
5.276.4 Member Function Documentation	1450
5.277 <code>__gnu_pbds::detail::lu_counter_metadata&lt; Size_Type &gt;</code> Class Template Reference	1451
5.277.1 Detailed Description	1451
5.278 <code>__gnu_pbds::detail::lu_counter_policy_base&lt; Size_Type &gt;</code> Class Template Reference	1452
5.278.1 Detailed Description	1452
5.279 <code>__gnu_pbds::detail::lu_map&lt; Key, Mapped, Eq_Fn, _Alloc, Update_Policy &gt;</code> Class Template Reference	1452
5.279.1 Detailed Description	1454
5.280 <code>__gnu_pbds::detail::mask_based_range_hashing&lt; Size_Type &gt;</code> Class Template Reference	1454
5.280.1 Detailed Description	1455
5.281 <code>__gnu_pbds::detail::mod_based_range_hashing&lt; Size_Type &gt;</code> Class Template Reference	1455
5.281.1 Detailed Description	1456
5.282 <code>__gnu_pbds::detail::no_throw_copies&lt; Key, Mapped &gt;</code> Struct Template Reference	1456
5.282.1 Detailed Description	1456
5.283 <code>__gnu_pbds::detail::no_throw_copies&lt; Key, null_type &gt;</code> Struct Template Reference	1457
5.283.1 Detailed Description	1457
5.284 <code>__gnu_pbds::detail::ov_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;</code> Class Template Reference	1457
5.284.1 Detailed Description	1459
5.284.2 Member Function Documentation	1459
5.285 <code>__gnu_pbds::detail::ov_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;::cond_dtor&lt; Size_Type &gt;</code> Class Template Reference	1461
5.285.1 Detailed Description	1461
5.286 <code>__gnu_pbds::detail::ov_tree_node_const_it&lt; Value_Type, Metadata_Type, _Alloc &gt;</code> Class Template Reference	1461
5.286.1 Detailed Description	1462
5.286.2 Member Function Documentation	1463
5.287 <code>__gnu_pbds::detail::ov_tree_node_it&lt; Value_Type, Metadata_Type, _Alloc &gt;</code> Class Template Reference	1463
5.287.1 Detailed Description	1464
5.287.2 Member Function Documentation	1464
5.288 <code>__gnu_pbds::detail::pairing_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt;</code> Class Template Reference	1465
5.288.1 Detailed Description	1467
5.289 <code>__gnu_pbds::detail::pat_trie_base</code> Struct Reference	1468
5.289.1 Detailed Description	1468
5.289.2 Member Enumeration Documentation	1468
5.290 <code>__gnu_pbds::detail::pat_trie_base::_CIter&lt; Node, Leaf, Head, Inode, Is_Forward_Iterator &gt;</code> Class Template Reference	1469
5.290.1 Detailed Description	1470

5.291	<a href="#">__gnu_pbds::detail::pat_trie_base::_Head&lt; _ATraits, Metadata &gt; Struct Template Reference</a>	1471
5.291.1	Detailed Description	1471
5.292	<a href="#">__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt; Struct Template Reference</a>	1472
5.292.1	Detailed Description	1473
5.293	<a href="#">__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt;::const_iterator Struct Reference</a>	1474
5.293.1	Detailed Description	1475
5.294	<a href="#">__gnu_pbds::detail::pat_trie_base::_Inode&lt; _ATraits, Metadata &gt;::iterator Struct Reference</a>	1475
5.294.1	Detailed Description	1476
5.295	<a href="#">__gnu_pbds::detail::pat_trie_base::_Iter&lt; Node, Leaf, Head, Inode, Is_Forward_Iterator &gt; Class Template Reference</a>	1476
5.295.1	Detailed Description	1478
5.296	<a href="#">__gnu_pbds::detail::pat_trie_base::_Leaf&lt; _ATraits, Metadata &gt; Struct Template Reference</a>	1478
5.296.1	Detailed Description	1479
5.297	<a href="#">__gnu_pbds::detail::pat_trie_base::_Metadata&lt; Metadata, _Alloc &gt; Struct Template Reference</a>	1479
5.297.1	Detailed Description	1480
5.298	<a href="#">__gnu_pbds::detail::pat_trie_base::_Metadata&lt; null_type, _Alloc &gt; Struct Template Reference</a>	1480
5.298.1	Detailed Description	1480
5.299	<a href="#">__gnu_pbds::detail::pat_trie_base::_Node_base&lt; _ATraits, Metadata &gt; Struct Template Reference</a>	1481
5.299.1	Detailed Description	1482
5.300	<a href="#">__gnu_pbds::detail::pat_trie_base::_Node_citer&lt; Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc &gt; Class Template Reference</a>	1482
5.300.1	Detailed Description	1483
5.300.2	Member Typedef Documentation	1483
5.300.3	Member Function Documentation	1484
5.301	<a href="#">__gnu_pbds::detail::pat_trie_base::_Node_iter&lt; Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc &gt; Class Template Reference</a>	1486
5.301.1	Detailed Description	1487
5.301.2	Member Typedef Documentation	1487
5.301.3	Member Function Documentation	1488
5.302	<a href="#">__gnu_pbds::detail::pat_trie_map&lt; Key, Mapped, Node_And_It_Traits, _Alloc &gt; Class Template Reference</a>	1490
5.302.1	Detailed Description	1492
5.302.2	Member Enumeration Documentation	1492
5.302.3	Member Function Documentation	1492
5.303	<a href="#">__gnu_pbds::detail::probe_fn_base&lt; _Alloc &gt; Class Template Reference</a>	1493
5.303.1	Detailed Description	1493
5.304	<a href="#">__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash &gt; Class Template Reference</a>	1494
5.304.1	Detailed Description	1494
5.305	<a href="#">__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false &gt; Class Template Reference</a>	1494

5.305.1 Detailed Description	1495
5.306 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true &gt;</code> Class Template Reference	1495
5.306.1 Detailed Description	1496
5.307 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, null_type, _Alloc, Comb_Hash_Fn, false &gt;</code> Class Template Reference	1496
5.307.1 Detailed Description	1496
5.308 <code>__gnu_pbds::detail::ranged_hash_fn&lt; Key, null_type, _Alloc, Comb_Hash_Fn, true &gt;</code> Class Template Reference	1497
5.308.1 Detailed Description	1497
5.309 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_↵ Hash &gt;</code> Class Template Reference	1497
5.309.1 Detailed Description	1498
5.310 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false &gt;</code> Class Template Reference	1498
5.310.1 Detailed Description	1498
5.311 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true &gt;</code> Class Template Reference	1499
5.311.1 Detailed Description	1499
5.312 <code>__gnu_pbds::detail::ranged_probe_fn&lt; Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false &gt;</code> Class Template Reference	1499
5.312.1 Detailed Description	1500
5.313 <code>__gnu_pbds::detail::rb_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;</code> Class Template Reference	1500
5.313.1 Detailed Description	1503
5.313.2 Member Function Documentation	1503
5.314 <code>__gnu_pbds::detail::rb_tree_node_&lt; Value_Type, Metadata, _Alloc &gt;</code> Struct Template Reference	1504
5.314.1 Detailed Description	1505
5.315 <code>__gnu_pbds::detail::rc&lt; _Node, _Alloc &gt;</code> Class Template Reference	1505
5.315.1 Detailed Description	1506
5.316 <code>__gnu_pbds::detail::rc_binomial_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt;</code> Class Template Reference	1506
5.316.1 Detailed Description	1508
5.317 <code>__gnu_pbds::detail::resize_policy&lt; _Tp &gt;</code> Class Template Reference	1508
5.317.1 Detailed Description	1509
5.318 <code>__gnu_pbds::detail::splay_tree_map&lt; Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc &gt;</code> Class Template Reference	1509
5.318.1 Detailed Description	1512
5.318.2 Member Function Documentation	1512
5.319 <code>__gnu_pbds::detail::splay_tree_node_&lt; Value_Type, Metadata, _Alloc &gt;</code> Struct Template Reference	1513
5.319.1 Detailed Description	1514
5.320 <code>__gnu_pbds::detail::stored_data&lt; _Tv, _Th &gt;</code> Struct Template Reference	1514
5.320.1 Detailed Description	1515

5.321	<a href="#">__gnu_pbds::detail::stored_data&lt; _Tv, null_type &gt; Struct Template Reference</a>	1515
5.321.1	Detailed Description	1516
5.322	<a href="#">__gnu_pbds::detail::stored_hash&lt; _Th &gt; Struct Template Reference</a>	1516
5.322.1	Detailed Description	1517
5.323	<a href="#">__gnu_pbds::detail::stored_value&lt; _Tv &gt; Struct Template Reference</a>	1517
5.323.1	Detailed Description	1517
5.324	<a href="#">__gnu_pbds::detail::synth_access_traits&lt; Type_Traits, Set, _ATraits &gt; Struct Template Reference</a>	1518
5.324.1	Detailed Description	1518
5.325	<a href="#">__gnu_pbds::detail::thin_heap&lt; Value_Type, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1518
5.325.1	Detailed Description	1520
5.326	<a href="#">__gnu_pbds::detail::tree_metadata_helper&lt; Node_Update, _BTp &gt; Struct Template Reference</a>	1521
5.326.1	Detailed Description	1521
5.327	<a href="#">__gnu_pbds::detail::tree_metadata_helper&lt; Node_Update, false &gt; Struct Template Reference</a>	1521
5.327.1	Detailed Description	1521
5.328	<a href="#">__gnu_pbds::detail::tree_metadata_helper&lt; Node_Update, true &gt; Struct Template Reference</a>	1521
5.328.1	Detailed Description	1522
5.329	<a href="#">__gnu_pbds::detail::tree_node_metadata_dispatch&lt; Key, Data, Cmp_Fn, Node_Update, _Alloc &gt; Struct Template Reference</a>	1522
5.329.1	Detailed Description	1522
5.330	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc &gt; Struct Template Reference</a>	1522
5.330.1	Detailed Description	1522
5.331	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc &gt; Struct Template Reference</a>	1523
5.331.1	Detailed Description	1523
5.331.2	Member Typedef Documentation	1523
5.332	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc &gt; Struct Template Reference</a>	1524
5.332.1	Detailed Description	1524
5.332.2	Member Typedef Documentation	1525
5.333	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc &gt; Struct Template Reference</a>	1525
5.333.1	Detailed Description	1526
5.333.2	Member Typedef Documentation	1526
5.334	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc &gt; Struct Template Reference</a>	1527
5.334.1	Detailed Description	1527
5.334.2	Member Typedef Documentation	1527
5.335	<a href="#">__gnu_pbds::detail::tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc &gt; Struct Template Reference</a>	1528
5.335.1	Detailed Description	1528

5.335.2 Member Typedef Documentation . . . . .	1529
5.336 <code>__gnu_pbds::detail::tree_traits&lt; Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc &gt;</code> Struct Template Reference . . . . .	1529
5.336.1 Detailed Description . . . . .	1530
5.336.2 Member Typedef Documentation . . . . .	1530
5.337 <code>__gnu_pbds::detail::trie_metadata_helper&lt; Node_Update, _BTp &gt;</code> Struct Template Reference . . . . .	1531
5.337.1 Detailed Description . . . . .	1531
5.338 <code>__gnu_pbds::detail::trie_metadata_helper&lt; Node_Update, false &gt;</code> Struct Template Reference . . . . .	1531
5.338.1 Detailed Description . . . . .	1531
5.339 <code>__gnu_pbds::detail::trie_metadata_helper&lt; Node_Update, true &gt;</code> Struct Template Reference . . . . .	1531
5.339.1 Detailed Description . . . . .	1532
5.340 <code>__gnu_pbds::detail::trie_node_metadata_dispatch&lt; Key, Data, Cmp_Fn, Node_Update, _Alloc &gt;</code> Struct Template Reference . . . . .	1532
5.340.1 Detailed Description . . . . .	1532
5.341 <code>__gnu_pbds::detail::trie_policy_base&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference . . . . .	1532
5.341.1 Detailed Description . . . . .	1534
5.342 <code>__gnu_pbds::detail::trie_traits&lt; Key, Data, _ATraits, Node_Update, Tag, _Alloc &gt;</code> Struct Template Reference . . . . .	1534
5.342.1 Detailed Description . . . . .	1534
5.343 <code>__gnu_pbds::detail::trie_traits&lt; Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc &gt;</code> Struct Template Reference . . . . .	1534
5.343.1 Detailed Description . . . . .	1535
5.343.2 Member Typedef Documentation . . . . .	1535
5.344 <code>__gnu_pbds::detail::trie_traits&lt; Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc &gt;</code> Struct Template Reference . . . . .	1536
5.344.1 Detailed Description . . . . .	1536
5.344.2 Member Typedef Documentation . . . . .	1536
5.345 <code>__gnu_pbds::detail::type_base&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference . . . . .	1537
5.345.1 Detailed Description . . . . .	1538
5.346 <code>__gnu_pbds::detail::type_base&lt; Key, Mapped, _Alloc, false &gt;</code> Struct Template Reference . . . . .	1538
5.346.1 Detailed Description . . . . .	1538
5.347 <code>__gnu_pbds::detail::type_base&lt; Key, Mapped, _Alloc, true &gt;</code> Struct Template Reference . . . . .	1539
5.347.1 Detailed Description . . . . .	1539
5.348 <code>__gnu_pbds::detail::type_base&lt; Key, null_type, _Alloc, false &gt;</code> Struct Template Reference . . . . .	1539
5.348.1 Detailed Description . . . . .	1540
5.349 <code>__gnu_pbds::detail::type_base&lt; Key, null_type, _Alloc, true &gt;</code> Struct Template Reference . . . . .	1540
5.349.1 Detailed Description . . . . .	1540
5.350 <code>__gnu_pbds::detail::type_dispatch&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference . . . . .	1541
5.350.1 Detailed Description . . . . .	1541
5.351 <code>__gnu_pbds::detail::types_traits&lt; Key, Mapped, _Alloc, Store_Hash &gt;</code> Struct Template Reference . . . . .	1541

5.351.1 Detailed Description . . . . .	1542
5.352 <a href="#">__gnu_pbds::direct_mask_range_hashing&lt; Size_Type &gt; Class Template Reference</a> . . . . .	1542
5.352.1 Detailed Description . . . . .	1543
5.352.2 Member Function Documentation . . . . .	1543
5.353 <a href="#">__gnu_pbds::direct_mod_range_hashing&lt; Size_Type &gt; Class Template Reference</a> . . . . .	1543
5.353.1 Detailed Description . . . . .	1544
5.353.2 Member Function Documentation . . . . .	1544
5.354 <a href="#">__gnu_pbds::gp_hash_table&lt; Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc &gt; Class Template Reference</a> . . . . .	1545
5.354.1 Detailed Description . . . . .	1546
5.354.2 Constructor & Destructor Documentation . . . . .	1546
5.355 <a href="#">__gnu_pbds::gp_hash_tag Struct Reference</a> . . . . .	1552
5.355.1 Detailed Description . . . . .	1552
5.356 <a href="#">__gnu_pbds::hash_exponential_size_policy&lt; Size_Type &gt; Class Template Reference</a> . . . . .	1552
5.356.1 Detailed Description . . . . .	1553
5.356.2 Constructor & Destructor Documentation . . . . .	1553
5.357 <a href="#">__gnu_pbds::hash_load_check_resize_trigger&lt; External_Load_Access, Size_Type &gt; Class Template Reference</a> . . . . .	1553
5.357.1 Detailed Description . . . . .	1554
5.357.2 Member Enumeration Documentation . . . . .	1554
5.357.3 Constructor & Destructor Documentation . . . . .	1555
5.357.4 Member Function Documentation . . . . .	1555
5.358 <a href="#">__gnu_pbds::hash_prime_size_policy Class Reference</a> . . . . .	1556
5.358.1 Detailed Description . . . . .	1557
5.358.2 Member Typedef Documentation . . . . .	1557
5.358.3 Constructor & Destructor Documentation . . . . .	1557
5.359 <a href="#">__gnu_pbds::hash_standard_resize_policy&lt; Size_Policy, Trigger_Policy, External_Size_Access, Size_Type &gt; Class Template Reference</a> . . . . .	1557
5.359.1 Detailed Description . . . . .	1558
5.359.2 Constructor & Destructor Documentation . . . . .	1559
5.359.3 Member Function Documentation . . . . .	1559
5.360 <a href="#">__gnu_pbds::insert_error Struct Reference</a> . . . . .	1562
5.360.1 Detailed Description . . . . .	1562
5.360.2 Member Function Documentation . . . . .	1562
5.361 <a href="#">__gnu_pbds::join_error Struct Reference</a> . . . . .	1563
5.361.1 Detailed Description . . . . .	1564
5.361.2 Member Function Documentation . . . . .	1564
5.362 <a href="#">__gnu_pbds::linear_probe_fn&lt; Size_Type &gt; Class Template Reference</a> . . . . .	1564
5.362.1 Detailed Description . . . . .	1564
5.362.2 Member Function Documentation . . . . .	1565

5.363	<a href="#">__gnu_pbds::list_update&lt; Key, Mapped, Eq_Fn, Update_Policy, _Alloc &gt; Class Template Reference</a>	1565
5.363.1	Detailed Description	1565
5.363.2	Constructor & Destructor Documentation	1566
5.364	<a href="#">__gnu_pbds::list_update_tag Struct Reference</a>	1567
5.364.1	Detailed Description	1567
5.365	<a href="#">__gnu_pbds::lu_counter_policy&lt; Max_Count, _Alloc &gt; Class Template Reference</a>	1567
5.365.1	Detailed Description	1568
5.365.2	Member Typedef Documentation	1568
5.365.3	Member Enumeration Documentation	1569
5.365.4	Member Function Documentation	1569
5.366	<a href="#">__gnu_pbds::lu_move_to_front_policy&lt; _Alloc &gt; Class Template Reference</a>	1570
5.366.1	Detailed Description	1570
5.366.2	Member Typedef Documentation	1570
5.366.3	Member Function Documentation	1571
5.367	<a href="#">__gnu_pbds::null_node_update&lt; _Tp1, _Tp2, _Tp3, _Tp4 &gt; Struct Template Reference</a>	1572
5.367.1	Detailed Description	1572
5.368	<a href="#">__gnu_pbds::null_type Struct Reference</a>	1572
5.368.1	Detailed Description	1573
5.369	<a href="#">__gnu_pbds::ov_tree_tag Struct Reference</a>	1573
5.369.1	Detailed Description	1574
5.370	<a href="#">__gnu_pbds::pairing_heap_tag Struct Reference</a>	1574
5.370.1	Detailed Description	1574
5.371	<a href="#">__gnu_pbds::pat_trie_tag Struct Reference</a>	1575
5.371.1	Detailed Description	1575
5.372	<a href="#">__gnu_pbds::point_invalidation_guarantee Struct Reference</a>	1576
5.372.1	Detailed Description	1576
5.373	<a href="#">__gnu_pbds::priority_queue&lt; _Tv, Cmp_Fn, Tag, _Alloc &gt; Class Template Reference</a>	1576
5.373.1	Detailed Description	1577
5.374	<a href="#">__gnu_pbds::priority_queue_tag Struct Reference</a>	1578
5.374.1	Detailed Description	1578
5.375	<a href="#">__gnu_pbds::quadratic_probe_fn&lt; Size_Type &gt; Class Template Reference</a>	1578
5.375.1	Detailed Description	1579
5.375.2	Member Function Documentation	1579
5.376	<a href="#">__gnu_pbds::range_invalidation_guarantee Struct Reference</a>	1579
5.376.1	Detailed Description	1580
5.377	<a href="#">__gnu_pbds::rb_tree_tag Struct Reference</a>	1580
5.377.1	Detailed Description	1580
5.378	<a href="#">__gnu_pbds::rc_binomial_heap_tag Struct Reference</a>	1581
5.378.1	Detailed Description	1581



5.379	<a href="#">__gnu_pbds::resize_error Struct Reference</a>	1582
5.379.1	Detailed Description	1582
5.379.2	Member Function Documentation	1582
5.380	<a href="#">__gnu_pbds::sample_probe_fn Class Reference</a>	1583
5.380.1	Detailed Description	1583
5.380.2	Constructor & Destructor Documentation	1583
5.380.3	Member Function Documentation	1584
5.381	<a href="#">__gnu_pbds::sample_range_hashing Class Reference</a>	1584
5.381.1	Detailed Description	1585
5.381.2	Member Typedef Documentation	1585
5.381.3	Constructor & Destructor Documentation	1585
5.381.4	Member Function Documentation	1585
5.382	<a href="#">__gnu_pbds::sample_ranged_hash_fn Class Reference</a>	1586
5.382.1	Detailed Description	1587
5.382.2	Constructor & Destructor Documentation	1587
5.382.3	Member Function Documentation	1587
5.383	<a href="#">__gnu_pbds::sample_ranged_probe_fn Class Reference</a>	1588
5.383.1	Detailed Description	1588
5.384	<a href="#">__gnu_pbds::sample_resize_policy Class Reference</a>	1588
5.384.1	Detailed Description	1589
5.384.2	Member Typedef Documentation	1589
5.384.3	Constructor & Destructor Documentation	1589
5.384.4	Member Function Documentation	1590
5.385	<a href="#">__gnu_pbds::sample_resize_trigger Class Reference</a>	1593
5.385.1	Detailed Description	1593
5.385.2	Member Typedef Documentation	1593
5.385.3	Constructor & Destructor Documentation	1594
5.385.4	Member Function Documentation	1594
5.386	<a href="#">__gnu_pbds::sample_size_policy Class Reference</a>	1597
5.386.1	Detailed Description	1597
5.386.2	Member Typedef Documentation	1597
5.386.3	Constructor & Destructor Documentation	1598
5.386.4	Member Function Documentation	1598
5.387	<a href="#">__gnu_pbds::sample_tree_node_update&lt; Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc &gt; Class Template Reference</a>	1599
5.387.1	Detailed Description	1599
5.388	<a href="#">__gnu_pbds::sample_trie_access_traits Struct Reference</a>	1599
5.388.1	Detailed Description	1600
5.388.2	Member Typedef Documentation	1600

5.388.3 Member Function Documentation	1600
5.389 <code>__gnu_pbds::sample_trie_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference	1601
5.389.1 Detailed Description	1601
5.389.2 Constructor & Destructor Documentation	1601
5.389.3 Member Function Documentation	1601
5.390 <code>__gnu_pbds::sample_update_policy</code> Struct Reference	1602
5.390.1 Detailed Description	1602
5.390.2 Member Typedef Documentation	1602
5.390.3 Constructor & Destructor Documentation	1602
5.390.4 Member Function Documentation	1603
5.391 <code>__gnu_pbds::sequence_tag</code> Struct Reference	1604
5.391.1 Detailed Description	1604
5.392 <code>__gnu_pbds::splay_tree_tag</code> Struct Reference	1605
5.392.1 Detailed Description	1605
5.393 <code>__gnu_pbds::string_tag</code> Struct Reference	1606
5.393.1 Detailed Description	1606
5.394 <code>__gnu_pbds::thin_heap_tag</code> Struct Reference	1607
5.394.1 Detailed Description	1607
5.395 <code>__gnu_pbds::tree&lt; Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc &gt;</code> Class Template Reference	1607
5.395.1 Detailed Description	1608
5.395.2 Member Typedef Documentation	1608
5.395.3 Constructor & Destructor Documentation	1609
5.396 <code>__gnu_pbds::tree_order_statistics_node_update&lt; Node_Cltr, Node_Itr, Cmp_Fn, _Alloc &gt;</code> Class Template Reference	1610
5.396.1 Detailed Description	1611
5.396.2 Member Function Documentation	1611
5.397 <code>__gnu_pbds::tree_tag</code> Struct Reference	1613
5.397.1 Detailed Description	1613
5.398 <code>__gnu_pbds::trie&lt; Key, Mapped, _ATraits, Tag, Node_Update, _Alloc &gt;</code> Class Template Reference	1613
5.398.1 Detailed Description	1614
5.398.2 Member Typedef Documentation	1614
5.398.3 Constructor & Destructor Documentation	1615
5.399 <code>__gnu_pbds::trie_order_statistics_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference	1616
5.399.1 Detailed Description	1618
5.399.2 Member Function Documentation	1618
5.400 <code>__gnu_pbds::trie_prefix_search_node_update&lt; Node_Cltr, Node_Itr, _ATraits, _Alloc &gt;</code> Class Template Reference	1620
5.400.1 Detailed Description	1621

5.400.2 Member Typedef Documentation . . . . .	1621
5.400.3 Member Function Documentation . . . . .	1622
5.401 <a href="#">__gnu_pbds::trie_string_access_traits&lt; String, Min_E_Val, Max_E_Val, Reverse, _Alloc &gt; Struct Template Reference</a> . . . . .	1624
5.401.1 Detailed Description . . . . .	1624
5.401.2 Member Typedef Documentation . . . . .	1625
5.401.3 Member Function Documentation . . . . .	1625
5.402 <a href="#">__gnu_pbds::trie_tag Struct Reference</a> . . . . .	1627
5.402.1 Detailed Description . . . . .	1627
5.403 <a href="#">__gnu_pbds::trivial_iterator_tag Struct Reference</a> . . . . .	1627
5.403.1 Detailed Description . . . . .	1627
5.404 <a href="#">__gnu_profile::__container_size_info Class Reference</a> . . . . .	1628
5.404.1 Detailed Description . . . . .	1629
5.405 <a href="#">__gnu_profile::__container_size_stack_info Class Reference</a> . . . . .	1629
5.405.1 Detailed Description . . . . .	1630
5.406 <a href="#">__gnu_profile::__hashfunc_info Class Reference</a> . . . . .	1630
5.406.1 Detailed Description . . . . .	1631
5.407 <a href="#">__gnu_profile::__hashfunc_stack_info Class Reference</a> . . . . .	1631
5.407.1 Detailed Description . . . . .	1632
5.408 <a href="#">__gnu_profile::__list2vector_info Class Reference</a> . . . . .	1632
5.408.1 Detailed Description . . . . .	1633
5.409 <a href="#">__gnu_profile::__map2umap_info Class Reference</a> . . . . .	1633
5.409.1 Detailed Description . . . . .	1634
5.410 <a href="#">__gnu_profile::__map2umap_stack_info Class Reference</a> . . . . .	1634
5.410.1 Detailed Description . . . . .	1635
5.411 <a href="#">__gnu_profile::__object_info_base Class Reference</a> . . . . .	1635
5.411.1 Detailed Description . . . . .	1636
5.412 <a href="#">__gnu_profile::__reentrance_guard Struct Reference</a> . . . . .	1636
5.412.1 Detailed Description . . . . .	1636
5.413 <a href="#">__gnu_profile::__stack_hash Class Reference</a> . . . . .	1637
5.413.1 Detailed Description . . . . .	1637
5.414 <a href="#">__gnu_profile::__trace_base&lt; __object_info, __stack_info &gt; Class Template Reference</a> . . . . .	1637
5.414.1 Detailed Description . . . . .	1637
5.415 <a href="#">__gnu_profile::__trace_container_size Class Reference</a> . . . . .	1638
5.415.1 Detailed Description . . . . .	1638
5.416 <a href="#">__gnu_profile::__trace_hash_func Class Reference</a> . . . . .	1639
5.416.1 Detailed Description . . . . .	1639
5.417 <a href="#">__gnu_profile::__trace_hashtable_size Class Reference</a> . . . . .	1640
5.417.1 Detailed Description . . . . .	1640

5.418 <a href="#">__gnu_profile::__trace_map2umap Class Reference</a>	1641
5.418.1 Detailed Description	1641
5.419 <a href="#">__gnu_profile::__trace_vector_size Class Reference</a>	1642
5.419.1 Detailed Description	1642
5.420 <a href="#">__gnu_profile::__trace_vector_to_list Class Reference</a>	1643
5.420.1 Detailed Description	1643
5.421 <a href="#">__gnu_profile::__vector2list_info Class Reference</a>	1644
5.421.1 Detailed Description	1645
5.422 <a href="#">__gnu_profile::__vector2list_stack_info Class Reference</a>	1645
5.422.1 Detailed Description	1646
5.423 <a href="#">__gnu_profile::__warning_data Struct Reference</a>	1646
5.423.1 Detailed Description	1646
5.424 <a href="#">const_iterator_ Class Reference</a>	1647
5.424.1 Detailed Description	1648
5.424.2 Member Typedef Documentation	1648
5.424.3 Constructor & Destructor Documentation	1650
5.424.4 Member Function Documentation	1650
5.424.5 Member Data Documentation	1652
5.425 <a href="#">iterator_ Class Reference</a>	1652
5.425.1 Detailed Description	1653
5.425.2 Member Typedef Documentation	1654
5.425.3 Constructor & Destructor Documentation	1655
5.425.4 Member Function Documentation	1655
5.425.5 Member Data Documentation	1658
5.426 <a href="#">point_const_iterator_ Class Reference</a>	1658
5.426.1 Detailed Description	1659
5.426.2 Member Typedef Documentation	1659
5.426.3 Constructor & Destructor Documentation	1661
5.426.4 Member Function Documentation	1661
5.427 <a href="#">point_iterator_ Class Reference</a>	1663
5.427.1 Detailed Description	1664
5.427.2 Member Typedef Documentation	1664
5.427.3 Constructor & Destructor Documentation	1665
5.427.4 Member Function Documentation	1666
5.428 <a href="#">std::__add_pointer_helper&lt; _Tp, bool &gt; Struct Template Reference</a>	1667
5.428.1 Detailed Description	1667
5.429 <a href="#">std::__allocated_ptr&lt; _Alloc &gt; Struct Template Reference</a>	1668
5.429.1 Detailed Description	1668
5.429.2 Constructor & Destructor Documentation	1668

5.429.3 Member Function Documentation . . . . .	1669
5.430 std::__atomic_base< _ITp > Struct Template Reference . . . . .	1670
5.430.1 Detailed Description . . . . .	1671
5.431 std::__atomic_base< _PTp * > Struct Template Reference . . . . .	1671
5.431.1 Detailed Description . . . . .	1671
5.432 std::__atomic_flag_base Struct Reference . . . . .	1672
5.432.1 Detailed Description . . . . .	1672
5.433 std::__basic_future< _Res > Class Template Reference . . . . .	1673
5.433.1 Detailed Description . . . . .	1674
5.433.2 Member Typedef Documentation . . . . .	1674
5.433.3 Member Function Documentation . . . . .	1674
5.434 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference . . . . .	1675
5.434.1 Detailed Description . . . . .	1676
5.434.2 Member Function Documentation . . . . .	1677
5.435 std::__ctype_abstract_base< _CharT > Class Template Reference . . . . .	1680
5.435.1 Detailed Description . . . . .	1682
5.435.2 Member Typedef Documentation . . . . .	1682
5.435.3 Member Function Documentation . . . . .	1682
5.436 std::__debug::bitset< _Nb > Class Template Reference . . . . .	1697
5.436.1 Detailed Description . . . . .	1698
5.437 std::__debug::deque< _Tp, _Allocator > Class Template Reference . . . . .	1699
5.437.1 Detailed Description . . . . .	1701
5.437.2 Member Function Documentation . . . . .	1701
5.437.3 Member Data Documentation . . . . .	1703
5.438 std::__debug::forward_list< _Tp, _Alloc > Class Template Reference . . . . .	1704
5.438.1 Detailed Description . . . . .	1706
5.438.2 Member Function Documentation . . . . .	1706
5.438.3 Member Data Documentation . . . . .	1708
5.439 std::__debug::list< _Tp, _Allocator > Class Template Reference . . . . .	1708
5.439.1 Detailed Description . . . . .	1711
5.439.2 Member Function Documentation . . . . .	1711
5.439.3 Member Data Documentation . . . . .	1713
5.440 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference . . . . .	1714
5.440.1 Detailed Description . . . . .	1716
5.440.2 Member Function Documentation . . . . .	1716
5.440.3 Member Data Documentation . . . . .	1718
5.441 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference . . . . .	1719
5.441.1 Detailed Description . . . . .	1722
5.441.2 Member Function Documentation . . . . .	1722

5.441.3 Member Data Documentation	1724
5.442 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	1724
5.442.1 Detailed Description	1727
5.442.2 Member Function Documentation	1727
5.442.3 Member Data Documentation	1729
5.443 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference	1730
5.443.1 Detailed Description	1732
5.443.2 Member Function Documentation	1732
5.443.3 Member Data Documentation	1734
5.444 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1735
5.444.1 Detailed Description	1738
5.444.2 Member Function Documentation	1738
5.444.3 Member Data Documentation	1740
5.445 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1741
5.445.1 Detailed Description	1744
5.445.2 Member Function Documentation	1744
5.445.3 Member Data Documentation	1746
5.446 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1747
5.446.1 Detailed Description	1749
5.446.2 Member Function Documentation	1750
5.446.3 Member Data Documentation	1752
5.447 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1753
5.447.1 Detailed Description	1755
5.447.2 Member Function Documentation	1756
5.447.3 Member Data Documentation	1758
5.448 std::__debug::vector< _Tp, _Allocator > Class Template Reference	1759
5.448.1 Detailed Description	1761
5.448.2 Constructor & Destructor Documentation	1762
5.448.3 Member Function Documentation	1762
5.448.4 Member Data Documentation	1764
5.449 std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference	1764
5.449.1 Detailed Description	1765
5.450 std::__detail::_Compiler< _TraitsT > Class Template Reference	1765
5.450.1 Detailed Description	1766
5.451 std::__detail::_Default_ranged_hash Struct Reference	1766
5.451.1 Detailed Description	1766
5.452 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_↵ code > Struct Template Reference	1766
5.452.1 Detailed Description	1766

5.453 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false > Struct Template Reference	1767
5.453.1 Detailed Description	1767
5.454 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true > Struct Template Reference	1767
5.454.1 Detailed Description	1767
5.455 std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1768
5.455.1 Detailed Description	1768
5.456 std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1769
5.456.1 Detailed Description	1769
5.457 std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1770
5.457.1 Detailed Description	1770
5.458 std::__detail::_Equality_base Struct Reference	1770
5.458.1 Detailed Description	1771
5.459 std::__detail::_Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode > Class Template Reference	1771
5.459.1 Detailed Description	1772
5.460 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference	1772
5.460.1 Detailed Description	1772
5.461 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false > Struct Template Reference	1773
5.461.1 Detailed Description	1774
5.462 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true > Struct Template Reference	1774
5.462.1 Detailed Description	1775
5.463 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference	1776
5.463.1 Detailed Description	1776
5.464 std::__detail::_Hash_node< _Value, _Cache_hash_code > Struct Template Reference	1777
5.464.1 Detailed Description	1777
5.465 std::__detail::_Hash_node< _Value, false > Struct Template Reference	1777
5.465.1 Detailed Description	1778
5.466 std::__detail::_Hash_node< _Value, true > Struct Template Reference	1778
5.466.1 Detailed Description	1779
5.467 std::__detail::_Hash_node_base Struct Reference	1779
5.467.1 Detailed Description	1780
5.468 std::__detail::_Hash_node_value_base< _Value > Struct Template Reference	1780
5.468.1 Detailed Description	1781
5.469 std::__detail::_Hashtable_alloc< _NodeAlloc > Struct Template Reference	1781

5.469.1 Detailed Description	1782
5.470 <code>std::__detail::_Hashtable_base&lt; _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits &gt; Struct Template Reference</code>	1782
5.470.1 Detailed Description	1783
5.471 <code>std::__detail::_Hashtable_ebo_helper&lt; _Nm, _Tp, __use_ebo &gt; Struct Template Reference</code>	1784
5.471.1 Detailed Description	1784
5.472 <code>std::__detail::_Hashtable_ebo_helper&lt; _Nm, _Tp, false &gt; Struct Template Reference</code>	1784
5.472.1 Detailed Description	1784
5.473 <code>std::__detail::_Hashtable_ebo_helper&lt; _Nm, _Tp, true &gt; Struct Template Reference</code>	1784
5.473.1 Detailed Description	1785
5.474 <code>std::__detail::_Hashtable_traits&lt; _Cache_hash_code, _Constant_iterators, _Unique_keys &gt; Struct Template Reference</code>	1785
5.474.1 Detailed Description	1785
5.475 <code>std::__detail::_Insert&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators &gt; Struct Template Reference</code>	1786
5.475.1 Detailed Description	1786
5.476 <code>std::__detail::_Insert&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false &gt; Struct Template Reference</code>	1787
5.476.1 Detailed Description	1788
5.477 <code>std::__detail::_Insert&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true &gt; Struct Template Reference</code>	1788
5.477.1 Detailed Description	1790
5.478 <code>std::__detail::_Insert_base&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits &gt; Struct Template Reference</code>	1790
5.478.1 Detailed Description	1791
5.479 <code>std::__detail::_List_node_base Struct Reference</code>	1792
5.479.1 Detailed Description	1792
5.480 <code>std::__detail::_List_node_header Struct Reference</code>	1793
5.480.1 Detailed Description	1793
5.481 <code>std::__detail::_Local_const_iterator&lt; _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache &gt; Struct Template Reference</code>	1794
5.481.1 Detailed Description	1794
5.482 <code>std::__detail::_Local_iterator&lt; _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache &gt; Struct Template Reference</code>	1795
5.482.1 Detailed Description	1795
5.483 <code>std::__detail::_Local_iterator_base&lt; _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code &gt; Struct Template Reference</code>	1796
5.483.1 Detailed Description	1796
5.484 <code>std::__detail::_Local_iterator_base&lt; _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true &gt; Struct Template Reference</code>	1796
5.484.1 Detailed Description	1797



5.485	<a href="#">std::__detail::_Map_base&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys &gt; Struct Template Reference</a>	1797
5.485.1	Detailed Description	1798
5.486	<a href="#">std::__detail::_Map_base&lt; _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false &gt; Struct Template Reference</a>	1798
5.486.1	Detailed Description	1798
5.487	<a href="#">std::__detail::_Map_base&lt; _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true &gt; Struct Template Reference</a>	1798
5.487.1	Detailed Description	1799
5.488	<a href="#">std::__detail::_Mask_range_hashing Struct Reference</a>	1799
5.488.1	Detailed Description	1799
5.489	<a href="#">std::__detail::_Mod_range_hashing Struct Reference</a>	1800
5.489.1	Detailed Description	1800
5.490	<a href="#">std::__detail::_Node_const_iterator&lt; _Value, __constant_iterators, __cache &gt; Struct Template Reference</a>	1800
5.490.1	Detailed Description	1801
5.491	<a href="#">std::__detail::_Node_iterator&lt; _Value, __constant_iterators, __cache &gt; Struct Template Reference</a>	1802
5.491.1	Detailed Description	1803
5.492	<a href="#">std::__detail::_Node_iterator_base&lt; _Value, _Cache_hash_code &gt; Struct Template Reference</a>	1803
5.492.1	Detailed Description	1803
5.493	<a href="#">std::__detail::_Power2_rehash_policy Struct Reference</a>	1804
5.493.1	Detailed Description	1804
5.494	<a href="#">std::__detail::_Prime_rehash_policy Struct Reference</a>	1804
5.494.1	Detailed Description	1805
5.495	<a href="#">std::__detail::_Quoted_string&lt; _String, _CharT &gt; Struct Template Reference</a>	1805
5.495.1	Detailed Description	1806
5.496	<a href="#">std::__detail::_Rehash_base&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename &gt; Struct Template Reference</a>	1806
5.496.1	Detailed Description	1806
5.497	<a href="#">std::__detail::_Rehash_base&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::false_type &gt; Struct Template Reference</a>	1807
5.497.1	Detailed Description	1807
5.498	<a href="#">std::__detail::_Rehash_base&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::true_type &gt; Struct Template Reference</a>	1807
5.498.1	Detailed Description	1807
5.499	<a href="#">std::__detail::_Scanner&lt; _CharT &gt; Class Template Reference</a>	1808
5.499.1	Detailed Description	1809
5.499.2	Member Enumeration Documentation	1809
5.500	<a href="#">std::__detail::_StateSeq&lt; _TraitsT &gt; Class Template Reference</a>	1810
5.500.1	Detailed Description	1810
5.501	<a href="#">std::__detector&lt; _Default, _AlwaysVoid, _Op, _Args &gt; Struct Template Reference</a>	1810
5.501.1	Detailed Description	1811

5.502	<a href="#">std::__detector&lt; _Default, __void_t&lt; _Op&lt; _Args... &gt; &gt;, _Op, _Args... &gt; Struct Template Reference</a>	1811
5.502.1	<a href="#">Detailed Description</a>	1811
5.503	<a href="#">std::__exception_ptr::exception_ptr Class Reference</a>	1811
5.503.1	<a href="#">Detailed Description</a>	1812
5.504	<a href="#">std::__future_base Struct Reference</a>	1812
5.504.1	<a href="#">Detailed Description</a>	1813
5.504.2	<a href="#">Member Typedef Documentation</a>	1813
5.505	<a href="#">std::__future_base::__Result&lt; _Res &gt; Struct Template Reference</a>	1814
5.505.1	<a href="#">Detailed Description</a>	1814
5.506	<a href="#">std::__future_base::__Result&lt; _Res &amp; &gt; Struct Template Reference</a>	1815
5.506.1	<a href="#">Detailed Description</a>	1815
5.507	<a href="#">std::__future_base::__Result&lt; void &gt; Struct Template Reference</a>	1816
5.507.1	<a href="#">Detailed Description</a>	1816
5.508	<a href="#">std::__future_base::__Result_alloc&lt; _Res, _Alloc &gt; Struct Template Reference</a>	1817
5.508.1	<a href="#">Detailed Description</a>	1817
5.509	<a href="#">std::__future_base::__Result_base Struct Reference</a>	1818
5.509.1	<a href="#">Detailed Description</a>	1818
5.510	<a href="#">std::__is_location_invariant&lt; _Tp &gt; Struct Template Reference</a>	1818
5.510.1	<a href="#">Detailed Description</a>	1819
5.511	<a href="#">std::__is_nullptr_t&lt; _Tp &gt; Struct Template Reference</a>	1819
5.511.1	<a href="#">Detailed Description</a>	1819
5.512	<a href="#">std::__is_trivially_copy_assignable_impl&lt; _Tp, bool &gt; Struct Template Reference</a>	1820
5.512.1	<a href="#">Detailed Description</a>	1820
5.513	<a href="#">std::__is_trivially_copy_constructible_impl&lt; _Tp, bool &gt; Struct Template Reference</a>	1820
5.513.1	<a href="#">Detailed Description</a>	1820
5.514	<a href="#">std::__is_trivially_move_assignable_impl&lt; _Tp, bool &gt; Struct Template Reference</a>	1820
5.514.1	<a href="#">Detailed Description</a>	1820
5.515	<a href="#">std::__is_trivially_move_constructible_impl&lt; _Tp, bool &gt; Struct Template Reference</a>	1821
5.515.1	<a href="#">Detailed Description</a>	1821
5.516	<a href="#">std::__is_tuple_like_impl&lt; std::pair&lt; _T1, _T2 &gt; &gt; Struct Template Reference</a>	1821
5.516.1	<a href="#">Detailed Description</a>	1822
5.517	<a href="#">std::__iterator_traits&lt; _Iterator, typename &gt; Struct Template Reference</a>	1822
5.517.1	<a href="#">Detailed Description</a>	1822
5.518	<a href="#">std::__numeric_limits_base Struct Reference</a>	1822
5.518.1	<a href="#">Detailed Description</a>	1823
5.518.2	<a href="#">Member Data Documentation</a>	1823
5.519	<a href="#">std::__parallel::__CRandNumber&lt; _MustBeInt &gt; Struct Template Reference</a>	1828
5.519.1	<a href="#">Detailed Description</a>	1828
5.520	<a href="#">std::__profile::bitset&lt; _Nb &gt; Class Template Reference</a>	1829

5.520.1 Detailed Description	1829
5.521 std::__profile::deque< _Tp, _Allocator > Class Template Reference	1830
5.521.1 Detailed Description	1830
5.522 std::__profile::forward_list< _Tp, _Alloc > Class Template Reference	1830
5.522.1 Detailed Description	1831
5.523 std::__profile::list< _Tp, _Allocator > Class Template Reference	1832
5.523.1 Detailed Description	1834
5.524 std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1834
5.524.1 Detailed Description	1836
5.525 std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1837
5.525.1 Detailed Description	1839
5.526 std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference	1839
5.526.1 Detailed Description	1842
5.527 std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference	1842
5.527.1 Detailed Description	1844
5.528 std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1845
5.528.1 Detailed Description	1846
5.529 std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1847
5.529.1 Detailed Description	1848
5.530 std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1848
5.530.1 Detailed Description	1850
5.531 std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference	1850
5.531.1 Detailed Description	1852
5.532 std::__shared_mutex_cv Class Reference	1852
5.532.1 Detailed Description	1853
5.533 std::_Base_bitset< _Nw > Struct Template Reference	1853
5.533.1 Detailed Description	1854
5.533.2 Member Data Documentation	1854
5.534 std::_Base_bitset< 0 > Struct Template Reference	1854
5.534.1 Detailed Description	1855
5.535 std::_Base_bitset< 1 > Struct Template Reference	1856
5.535.1 Detailed Description	1857
5.536 std::_Bind< _Signature > Struct Template Reference	1857
5.536.1 Detailed Description	1857
5.537 std::_Bind_result< _Result, _Signature > Struct Template Reference	1857
5.537.1 Detailed Description	1857
5.538 std::_Deque_base< _Tp, _Alloc > Class Template Reference	1858
5.538.1 Detailed Description	1859
5.538.2 Member Function Documentation	1859

5.539 <a href="#">std::_Deque_iterator&lt; _Tp, _Ref, _Ptr &gt; Struct Template Reference</a>	1860
5.539.1 Detailed Description	1861
5.539.2 Member Function Documentation	1861
5.540 <a href="#">std::_Enable_copy_move&lt; _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag &gt; Struct Template Reference</a>	1862
5.540.1 Detailed Description	1862
5.541 <a href="#">std::_Enable_default_constructor&lt; _Switch, _Tag &gt; Struct Template Reference</a>	1862
5.541.1 Detailed Description	1863
5.542 <a href="#">std::_Enable_destructor&lt; _Switch, _Tag &gt; Struct Template Reference</a>	1863
5.542.1 Detailed Description	1863
5.543 <a href="#">std::_Enable_special_members&lt; _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag &gt; Struct Template Reference</a>	1864
5.543.1 Detailed Description	1864
5.544 <a href="#">std::_Function_base Class Reference</a>	1865
5.544.1 Detailed Description	1865
5.545 <a href="#">std::_Fwd_list_base&lt; _Tp, _Alloc &gt; Struct Template Reference</a>	1866
5.545.1 Detailed Description	1867
5.546 <a href="#">std::_Fwd_list_const_iterator&lt; _Tp &gt; Struct Template Reference</a>	1867
5.546.1 Detailed Description	1868
5.547 <a href="#">std::_Fwd_list_iterator&lt; _Tp &gt; Struct Template Reference</a>	1868
5.547.1 Detailed Description	1869
5.548 <a href="#">std::_Fwd_list_node&lt; _Tp &gt; Struct Template Reference</a>	1869
5.548.1 Detailed Description	1870
5.549 <a href="#">std::_Fwd_list_node_base Struct Reference</a>	1870
5.549.1 Detailed Description	1871
5.550 <a href="#">std::_Hashtable&lt; _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits &gt; Class Template Reference</a>	1871
5.550.1 Detailed Description	1875
5.551 <a href="#">std::_List_base&lt; _Tp, _Alloc &gt; Class Template Reference</a>	1877
5.551.1 Detailed Description	1878
5.552 <a href="#">std::_List_const_iterator&lt; _Tp &gt; Struct Template Reference</a>	1878
5.552.1 Detailed Description	1879
5.553 <a href="#">std::_List_iterator&lt; _Tp &gt; Struct Template Reference</a>	1879
5.553.1 Detailed Description	1880
5.554 <a href="#">std::_List_node&lt; _Tp &gt; Struct Template Reference</a>	1880
5.554.1 Detailed Description	1881
5.555 <a href="#">std::_Maybe_get_result_type&lt; _Functor, typename &gt; Struct Template Reference</a>	1881
5.555.1 Detailed Description	1881
5.556 <a href="#">std::_Maybe_unary_or_binary_function&lt; _Res, _ArgTypes &gt; Struct Template Reference</a>	1882
5.556.1 Detailed Description	1882

5.557 <a href="#">std::_Maybe_unary_or_binary_function&lt;_Res, _T1 &gt; Struct Template Reference</a>	1882
5.557.1 Detailed Description	1882
5.557.2 Member Typedef Documentation	1883
5.558 <a href="#">std::_Maybe_unary_or_binary_function&lt;_Res, _T1, _T2 &gt; Struct Template Reference</a>	1883
5.558.1 Detailed Description	1884
5.558.2 Member Typedef Documentation	1884
5.559 <a href="#">std::_Mu&lt;_Arg, _IsBindExp, _IsPlaceholder &gt; Class Template Reference</a>	1885
5.559.1 Detailed Description	1885
5.560 <a href="#">std::_Mu&lt;_Arg, false, false &gt; Class Template Reference</a>	1885
5.560.1 Detailed Description	1885
5.561 <a href="#">std::_Mu&lt;_Arg, false, true &gt; Class Template Reference</a>	1885
5.561.1 Detailed Description	1886
5.562 <a href="#">std::_Mu&lt;_Arg, true, false &gt; Class Template Reference</a>	1886
5.562.1 Detailed Description	1886
5.563 <a href="#">std::_Mu&lt;reference_wrapper&lt;_Tp &gt;, false, false &gt; Class Template Reference</a>	1886
5.563.1 Detailed Description	1887
5.564 <a href="#">std::_Not_fn&lt;_Fn &gt; Class Template Reference</a>	1887
5.564.1 Detailed Description	1887
5.565 <a href="#">std::_Placeholder&lt;_Num &gt; Struct Template Reference</a>	1888
5.565.1 Detailed Description	1888
5.566 <a href="#">std::_Reference_wrapper_base&lt;_Tp &gt; Struct Template Reference</a>	1888
5.566.1 Detailed Description	1888
5.567 <a href="#">std::_Sp_ebo_helper&lt;_Nm, _Tp, false &gt; Struct Template Reference</a>	1889
5.567.1 Detailed Description	1889
5.568 <a href="#">std::_Sp_ebo_helper&lt;_Nm, _Tp, true &gt; Struct Template Reference</a>	1889
5.568.1 Detailed Description	1889
5.569 <a href="#">std::_Temporary_buffer&lt;_ForwardIterator, _Tp &gt; Class Template Reference</a>	1890
5.569.1 Detailed Description	1890
5.569.2 Constructor & Destructor Documentation	1891
5.569.3 Member Function Documentation	1891
5.570 <a href="#">std::_Tuple_impl&lt;_Idx, _Elements &gt; Struct Template Reference</a>	1892
5.570.1 Detailed Description	1892
5.571 <a href="#">std::_Tuple_impl&lt;_Idx, _Head, _Tail... &gt; Struct Template Reference</a>	1892
5.571.1 Detailed Description	1894
5.572 <a href="#">std::_V2::condition_variable_any Class Reference</a>	1894
5.572.1 Detailed Description	1894
5.573 <a href="#">std::_V2::error_category Class Reference</a>	1895
5.573.1 Detailed Description	1895
5.574 <a href="#">std::_Vector_base&lt;_Tp, _Alloc &gt; Struct Template Reference</a>	1895

5.574.1 Detailed Description . . . . .	1896
5.575 std::_Weak_result_type< _Functor > Struct Template Reference . . . . .	1896
5.575.1 Detailed Description . . . . .	1897
5.576 std::_Weak_result_type_impl< _Functor > Struct Template Reference . . . . .	1897
5.576.1 Detailed Description . . . . .	1897
5.577 std::_Weak_result_type_impl< _Res(*)(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL > Struct Template Reference . . . . .	1898
5.577.1 Detailed Description . . . . .	1898
5.578 std::_Weak_result_type_impl< _Res(*)(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL > Struct Template Reference . . . . .	1898
5.578.1 Detailed Description . . . . .	1898
5.579 std::_Weak_result_type_impl< _Res(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL > Struct Template Reference . . . . .	1898
5.579.1 Detailed Description . . . . .	1899
5.580 std::_Weak_result_type_impl< _Res(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL > Struct Template Reference . . . . .	1899
5.580.1 Detailed Description . . . . .	1899
5.581 std::add_const< _Tp > Struct Template Reference . . . . .	1899
5.581.1 Detailed Description . . . . .	1900
5.582 std::add_cv< _Tp > Struct Template Reference . . . . .	1900
5.582.1 Detailed Description . . . . .	1900
5.583 std::add_lvalue_reference< _Tp > Struct Template Reference . . . . .	1900
5.583.1 Detailed Description . . . . .	1901
5.584 std::add_rvalue_reference< _Tp > Struct Template Reference . . . . .	1901
5.584.1 Detailed Description . . . . .	1901
5.585 std::add_volatile< _Tp > Struct Template Reference . . . . .	1901
5.585.1 Detailed Description . . . . .	1902
5.586 std::adopt_lock_t Struct Reference . . . . .	1902
5.586.1 Detailed Description . . . . .	1902
5.587 std::aligned_storage< _Len, _Align > Struct Template Reference . . . . .	1902
5.587.1 Detailed Description . . . . .	1902
5.588 std::aligned_union< _Len, _Types > Struct Template Reference . . . . .	1903
5.588.1 Detailed Description . . . . .	1903
5.588.2 Member Typedef Documentation . . . . .	1903
5.589 std::alignment_of< _Tp > Struct Template Reference . . . . .	1904
5.589.1 Detailed Description . . . . .	1904
5.590 std::allocator< _Tp > Class Template Reference . . . . .	1905
5.590.1 Detailed Description . . . . .	1906
5.591 std::allocator< void > Class Template Reference . . . . .	1907
5.591.1 Detailed Description . . . . .	1907

5.592 std::allocator_arg_t Struct Reference . . . . .	1908
5.592.1 Detailed Description . . . . .	1908
5.593 std::allocator_traits< _Alloc > Struct Template Reference . . . . .	1908
5.593.1 Detailed Description . . . . .	1909
5.593.2 Member Typedef Documentation . . . . .	1909
5.593.3 Member Function Documentation . . . . .	1912
5.594 std::allocator_traits< allocator< _Tp > > Struct Template Reference . . . . .	1916
5.594.1 Detailed Description . . . . .	1917
5.594.2 Member Typedef Documentation . . . . .	1917
5.594.3 Member Function Documentation . . . . .	1919
5.595 std::array< _Tp, _Nm > Struct Template Reference . . . . .	1923
5.595.1 Detailed Description . . . . .	1924
5.596 std::atomic< _Tp > Struct Template Reference . . . . .	1924
5.596.1 Detailed Description . . . . .	1925
5.597 std::atomic< _Tp * > Struct Template Reference . . . . .	1925
5.597.1 Detailed Description . . . . .	1927
5.598 std::atomic< bool > Struct Template Reference . . . . .	1927
5.598.1 Detailed Description . . . . .	1928
5.599 std::atomic< char > Struct Template Reference . . . . .	1928
5.599.1 Detailed Description . . . . .	1929
5.600 std::atomic< char16_t > Struct Template Reference . . . . .	1929
5.600.1 Detailed Description . . . . .	1930
5.601 std::atomic< char32_t > Struct Template Reference . . . . .	1931
5.601.1 Detailed Description . . . . .	1932
5.602 std::atomic< int > Struct Template Reference . . . . .	1932
5.602.1 Detailed Description . . . . .	1933
5.603 std::atomic< long > Struct Template Reference . . . . .	1933
5.603.1 Detailed Description . . . . .	1934
5.604 std::atomic< long long > Struct Template Reference . . . . .	1935
5.604.1 Detailed Description . . . . .	1936
5.605 std::atomic< short > Struct Template Reference . . . . .	1936
5.605.1 Detailed Description . . . . .	1937
5.606 std::atomic< signed char > Struct Template Reference . . . . .	1937
5.606.1 Detailed Description . . . . .	1938
5.607 std::atomic< unsigned char > Struct Template Reference . . . . .	1939
5.607.1 Detailed Description . . . . .	1940
5.608 std::atomic< unsigned int > Struct Template Reference . . . . .	1940
5.608.1 Detailed Description . . . . .	1941
5.609 std::atomic< unsigned long > Struct Template Reference . . . . .	1941

5.609.1 Detailed Description	1942
5.610 <code>std::atomic&lt; unsigned long long &gt;</code> Struct Template Reference	1943
5.610.1 Detailed Description	1944
5.611 <code>std::atomic&lt; unsigned short &gt;</code> Struct Template Reference	1944
5.611.1 Detailed Description	1945
5.612 <code>std::atomic&lt; wchar_t &gt;</code> Struct Template Reference	1945
5.612.1 Detailed Description	1946
5.613 <code>std::atomic_flag</code> Struct Reference	1947
5.613.1 Detailed Description	1947
5.614 <code>std::auto_ptr&lt; _Tp &gt;</code> Class Template Reference	1947
5.614.1 Detailed Description	1948
5.614.2 Member Typedef Documentation	1949
5.614.3 Constructor & Destructor Documentation	1949
5.614.4 Member Function Documentation	1951
5.615 <code>std::auto_ptr_ref&lt; _Tp1 &gt;</code> Struct Template Reference	1953
5.615.1 Detailed Description	1954
5.616 <code>std::back_insert_iterator&lt; _Container &gt;</code> Class Template Reference	1954
5.616.1 Detailed Description	1955
5.616.2 Member Typedef Documentation	1955
5.616.3 Constructor & Destructor Documentation	1956
5.616.4 Member Function Documentation	1957
5.617 <code>std::bad_alloc</code> Class Reference	1958
5.617.1 Detailed Description	1958
5.617.2 Member Function Documentation	1958
5.618 <code>std::bad_cast</code> Class Reference	1959
5.618.1 Detailed Description	1959
5.618.2 Member Function Documentation	1960
5.619 <code>std::bad_exception</code> Class Reference	1960
5.619.1 Detailed Description	1960
5.619.2 Member Function Documentation	1961
5.620 <code>std::bad_function_call</code> Class Reference	1961
5.620.1 Detailed Description	1961
5.620.2 Member Function Documentation	1962
5.621 <code>std::bad_typeid</code> Class Reference	1962
5.621.1 Detailed Description	1962
5.621.2 Member Function Documentation	1963
5.622 <code>std::bad_weak_ptr</code> Class Reference	1963
5.622.1 Detailed Description	1963
5.622.2 Member Function Documentation	1964



---

5.623 <code>std::basic_filebuf&lt; _CharT, _Traits &gt;</code> Class Template Reference	1964
5.623.1 Detailed Description	1967
5.623.2 Constructor & Destructor Documentation	1968
5.623.3 Member Function Documentation	1968
5.623.4 Member Data Documentation	1988
5.624 <code>std::basic_fstream&lt; _CharT, _Traits &gt;</code> Class Template Reference	1993
5.624.1 Detailed Description	1999
5.624.2 Member Typedef Documentation	2000
5.624.3 Member Enumeration Documentation	2002
5.624.4 Constructor & Destructor Documentation	2003
5.624.5 Member Function Documentation	2004
5.624.6 Member Data Documentation	2052
5.625 <code>std::basic_ifstream&lt; _CharT, _Traits &gt;</code> Class Template Reference	2060
5.625.1 Detailed Description	2065
5.625.2 Member Typedef Documentation	2065
5.625.3 Member Enumeration Documentation	2068
5.625.4 Constructor & Destructor Documentation	2069
5.625.5 Member Function Documentation	2070
5.625.6 Member Data Documentation	2106
5.626 <code>std::basic_ios&lt; _CharT, _Traits &gt;</code> Class Template Reference	2114
5.626.1 Detailed Description	2117
5.626.2 Member Typedef Documentation	2117
5.626.3 Member Enumeration Documentation	2122
5.626.4 Constructor & Destructor Documentation	2122
5.626.5 Member Function Documentation	2123
5.626.6 Member Data Documentation	2138
5.627 <code>std::basic_iostream&lt; _CharT, _Traits &gt;</code> Class Template Reference	2146
5.627.1 Detailed Description	2152
5.627.2 Member Typedef Documentation	2153
5.627.3 Member Enumeration Documentation	2155
5.627.4 Constructor & Destructor Documentation	2156
5.627.5 Member Function Documentation	2156
5.627.6 Member Data Documentation	2203
5.628 <code>std::basic_istream&lt; _CharT, _Traits &gt;</code> Class Template Reference	2210
5.628.1 Detailed Description	2216
5.628.2 Member Typedef Documentation	2216
5.628.3 Member Enumeration Documentation	2219
5.628.4 Constructor & Destructor Documentation	2219
5.628.5 Member Function Documentation	2220

---

5.628.6 Member Data Documentation . . . . .	2255
5.629 std::basic_istream< _CharT, _Traits >::sentry Class Reference . . . . .	2263
5.629.1 Detailed Description . . . . .	2263
5.629.2 Member Typedef Documentation . . . . .	2263
5.629.3 Constructor & Destructor Documentation . . . . .	2263
5.629.4 Member Function Documentation . . . . .	2264
5.630 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	2265
5.630.1 Detailed Description . . . . .	2270
5.630.2 Member Typedef Documentation . . . . .	2270
5.630.3 Member Enumeration Documentation . . . . .	2273
5.630.4 Constructor & Destructor Documentation . . . . .	2274
5.630.5 Member Function Documentation . . . . .	2275
5.630.6 Member Data Documentation . . . . .	2311
5.631 std::basic_ofstream< _CharT, _Traits > Class Template Reference . . . . .	2319
5.631.1 Detailed Description . . . . .	2323
5.631.2 Member Typedef Documentation . . . . .	2324
5.631.3 Member Enumeration Documentation . . . . .	2326
5.631.4 Constructor & Destructor Documentation . . . . .	2327
5.631.5 Member Function Documentation . . . . .	2328
5.631.6 Member Data Documentation . . . . .	2355
5.632 std::basic_ostream< _CharT, _Traits > Class Template Reference . . . . .	2363
5.632.1 Detailed Description . . . . .	2367
5.632.2 Member Typedef Documentation . . . . .	2368
5.632.3 Member Enumeration Documentation . . . . .	2370
5.632.4 Constructor & Destructor Documentation . . . . .	2371
5.632.5 Member Function Documentation . . . . .	2371
5.632.6 Member Data Documentation . . . . .	2398
5.633 std::basic_ostream< _CharT, _Traits >::sentry Class Reference . . . . .	2405
5.633.1 Detailed Description . . . . .	2405
5.633.2 Constructor & Destructor Documentation . . . . .	2405
5.633.3 Member Function Documentation . . . . .	2406
5.634 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . . .	2407
5.634.1 Detailed Description . . . . .	2411
5.634.2 Member Typedef Documentation . . . . .	2412
5.634.3 Member Enumeration Documentation . . . . .	2414
5.634.4 Constructor & Destructor Documentation . . . . .	2415
5.634.5 Member Function Documentation . . . . .	2416
5.634.6 Member Data Documentation . . . . .	2443
5.635 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference . . . . .	2450

5.635.1 Detailed Description	2452
5.635.2 Constructor & Destructor Documentation	2452
5.635.3 Member Function Documentation	2456
5.636 std::basic_streambuf<_CharT, _Traits> Class Template Reference	2463
5.636.1 Detailed Description	2465
5.636.2 Member Typedef Documentation	2467
5.636.3 Constructor & Destructor Documentation	2468
5.636.4 Member Function Documentation	2469
5.636.5 Member Data Documentation	2484
5.637 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference	2486
5.637.1 Detailed Description	2490
5.637.2 Constructor & Destructor Documentation	2490
5.637.3 Member Function Documentation	2495
5.637.4 Member Data Documentation	2551
5.638 std::basic_stringbuf<_CharT, _Traits, _Alloc> Class Template Reference	2552
5.638.1 Detailed Description	2554
5.638.2 Constructor & Destructor Documentation	2554
5.638.3 Member Function Documentation	2555
5.638.4 Member Data Documentation	2573
5.639 std::basic_stringstream<_CharT, _Traits, _Alloc> Class Template Reference	2575
5.639.1 Detailed Description	2581
5.639.2 Member Typedef Documentation	2582
5.639.3 Member Enumeration Documentation	2585
5.639.4 Constructor & Destructor Documentation	2585
5.639.5 Member Function Documentation	2586
5.639.6 Member Data Documentation	2633
5.640 std::bernoulli_distribution Class Reference	2640
5.640.1 Detailed Description	2641
5.640.2 Member Typedef Documentation	2641
5.640.3 Constructor & Destructor Documentation	2642
5.640.4 Member Function Documentation	2642
5.640.5 Friends And Related Function Documentation	2644
5.641 std::bernoulli_distribution::param_type Struct Reference	2644
5.641.1 Detailed Description	2645
5.642 std::bidirectional_iterator_tag Struct Reference	2645
5.642.1 Detailed Description	2645
5.643 std::binary_function<_Arg1, _Arg2, _Result> Struct Template Reference	2646
5.643.1 Detailed Description	2646
5.643.2 Member Typedef Documentation	2646

5.644 <a href="#">std::binary_negate&lt; _Predicate &gt; Class Template Reference</a>	2647
5.644.1 Detailed Description	2648
5.644.2 Member Typedef Documentation	2648
5.645 <a href="#">std::binder1st&lt; _Operation &gt; Class Template Reference</a>	2649
5.645.1 Detailed Description	2650
5.645.2 Member Typedef Documentation	2650
5.646 <a href="#">std::binder2nd&lt; _Operation &gt; Class Template Reference</a>	2650
5.646.1 Detailed Description	2651
5.646.2 Member Typedef Documentation	2651
5.647 <a href="#">std::binomial_distribution&lt; _IntType &gt; Class Template Reference</a>	2652
5.647.1 Detailed Description	2653
5.647.2 Member Typedef Documentation	2653
5.647.3 Member Function Documentation	2653
5.647.4 Friends And Related Function Documentation	2656
5.648 <a href="#">std::binomial_distribution&lt; _IntType &gt;::param_type Struct Reference</a>	2657
5.648.1 Detailed Description	2657
5.649 <a href="#">std::bitset&lt; _Nb &gt; Class Template Reference</a>	2658
5.649.1 Detailed Description	2661
5.649.2 Constructor & Destructor Documentation	2662
5.649.3 Member Function Documentation	2664
5.650 <a href="#">std::bitset&lt; _Nb &gt;::reference Class Reference</a>	2673
5.650.1 Detailed Description	2673
5.651 <a href="#">std::cauchy_distribution&lt; _RealType &gt; Class Template Reference</a>	2674
5.651.1 Detailed Description	2674
5.651.2 Member Typedef Documentation	2675
5.651.3 Member Function Documentation	2675
5.651.4 Friends And Related Function Documentation	2676
5.652 <a href="#">std::cauchy_distribution&lt; _RealType &gt;::param_type Struct Reference</a>	2677
5.652.1 Detailed Description	2677
5.653 <a href="#">std::char_traits&lt; _CharT &gt; Struct Template Reference</a>	2677
5.653.1 Detailed Description	2678
5.654 <a href="#">std::char_traits&lt; __gnu_cxx::character&lt; _Value, _Int, _St &gt; &gt; Struct Template Reference</a>	2679
5.654.1 Detailed Description	2679
5.655 <a href="#">std::char_traits&lt; char &gt; Struct Template Reference</a>	2679
5.655.1 Detailed Description	2680
5.656 <a href="#">std::char_traits&lt; wchar_t &gt; Struct Template Reference</a>	2680
5.656.1 Detailed Description	2681
5.657 <a href="#">std::chi_squared_distribution&lt; _RealType &gt; Class Template Reference</a>	2681
5.657.1 Detailed Description	2682

5.657.2 Member Typedef Documentation . . . . .	2682
5.657.3 Member Function Documentation . . . . .	2683
5.657.4 Friends And Related Function Documentation . . . . .	2684
5.658 std::chi_squared_distribution< _RealType >::param_type Struct Reference . . . . .	2686
5.658.1 Detailed Description . . . . .	2686
5.659 std::chrono::_V2::steady_clock Struct Reference . . . . .	2686
5.659.1 Detailed Description . . . . .	2687
5.660 std::chrono::_V2::system_clock Struct Reference . . . . .	2687
5.660.1 Detailed Description . . . . .	2687
5.661 std::chrono::duration< _Rep, _Period > Struct Template Reference . . . . .	2688
5.661.1 Detailed Description . . . . .	2688
5.662 std::chrono::duration_values< _Rep > Struct Template Reference . . . . .	2689
5.662.1 Detailed Description . . . . .	2689
5.663 std::chrono::time_point< _Clock, _Dur > Struct Template Reference . . . . .	2689
5.663.1 Detailed Description . . . . .	2690
5.664 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference . . . . .	2690
5.664.1 Detailed Description . . . . .	2690
5.665 std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference . . . . .	2691
5.665.1 Detailed Description . . . . .	2692
5.665.2 Member Function Documentation . . . . .	2692
5.666 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference . . . . .	2696
5.666.1 Detailed Description . . . . .	2697
5.666.2 Member Function Documentation . . . . .	2698
5.667 std::codecvt< char, char, mbstate_t > Class Template Reference . . . . .	2701
5.667.1 Detailed Description . . . . .	2702
5.667.2 Member Function Documentation . . . . .	2702
5.668 std::codecvt< char16_t, char, mbstate_t > Class Template Reference . . . . .	2706
5.668.1 Detailed Description . . . . .	2707
5.668.2 Member Function Documentation . . . . .	2707
5.669 std::codecvt< char32_t, char, mbstate_t > Class Template Reference . . . . .	2711
5.669.1 Detailed Description . . . . .	2712
5.669.2 Member Function Documentation . . . . .	2712
5.670 std::codecvt< wchar_t, char, mbstate_t > Class Template Reference . . . . .	2716
5.670.1 Detailed Description . . . . .	2717
5.670.2 Member Function Documentation . . . . .	2717
5.671 std::codecvt_base Class Reference . . . . .	2721
5.671.1 Detailed Description . . . . .	2721
5.672 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference . . . . .	2722
5.672.1 Detailed Description . . . . .	2723

5.672.2 Member Function Documentation	2723
5.673 std::collate< _CharT > Class Template Reference	2727
5.673.1 Detailed Description	2729
5.673.2 Member Typedef Documentation	2729
5.673.3 Constructor & Destructor Documentation	2729
5.673.4 Member Function Documentation	2730
5.673.5 Member Data Documentation	2734
5.674 std::collate_byname< _CharT > Class Template Reference	2734
5.674.1 Detailed Description	2736
5.674.2 Member Typedef Documentation	2736
5.674.3 Member Function Documentation	2736
5.674.4 Member Data Documentation	2740
5.675 std::common_type< _Tp > Struct Template Reference	2740
5.675.1 Detailed Description	2740
5.676 std::complex< _Tp > Struct Template Reference	2741
5.676.1 Detailed Description	2741
5.676.2 Member Typedef Documentation	2742
5.676.3 Constructor & Destructor Documentation	2742
5.676.4 Member Function Documentation	2742
5.677 std::complex< double > Struct Template Reference	2743
5.677.1 Detailed Description	2744
5.678 std::complex< float > Struct Template Reference	2744
5.678.1 Detailed Description	2745
5.679 std::complex< long double > Struct Template Reference	2745
5.679.1 Detailed Description	2746
5.680 std::condition_variable Class Reference	2746
5.680.1 Detailed Description	2747
5.681 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference	2747
5.681.1 Detailed Description	2747
5.682 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2747
5.682.1 Detailed Description	2748
5.682.2 Member Typedef Documentation	2748
5.683 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2749
5.683.1 Detailed Description	2750
5.683.2 Member Typedef Documentation	2750
5.684 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2751
5.684.1 Detailed Description	2751
5.684.2 Member Typedef Documentation	2751
5.685 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference	2752

5.685.1 Detailed Description	2753
5.685.2 Member Typedef Documentation	2753
5.686 std::ctype< _CharT > Class Template Reference	2754
5.686.1 Detailed Description	2756
5.686.2 Member Function Documentation	2756
5.686.3 Member Data Documentation	2771
5.687 std::ctype< char > Class Template Reference	2772
5.687.1 Detailed Description	2774
5.687.2 Member Typedef Documentation	2774
5.687.3 Constructor & Destructor Documentation	2774
5.687.4 Member Function Documentation	2775
5.687.5 Member Data Documentation	2788
5.688 std::ctype< wchar_t > Class Template Reference	2789
5.688.1 Detailed Description	2791
5.688.2 Member Typedef Documentation	2791
5.688.3 Constructor & Destructor Documentation	2791
5.688.4 Member Function Documentation	2792
5.688.5 Member Data Documentation	2806
5.689 std::ctype_base Struct Reference	2806
5.689.1 Detailed Description	2807
5.690 std::ctype_byname< _CharT > Class Template Reference	2807
5.690.1 Detailed Description	2809
5.690.2 Member Function Documentation	2809
5.690.3 Member Data Documentation	2824
5.691 std::ctype_byname< char > Class Template Reference	2825
5.691.1 Detailed Description	2827
5.691.2 Member Typedef Documentation	2827
5.691.3 Member Function Documentation	2827
5.691.4 Member Data Documentation	2841
5.692 std::decay< _Tp > Class Template Reference	2841
5.692.1 Detailed Description	2842
5.693 std::decimal::decimal128 Class Reference	2842
5.693.1 Detailed Description	2843
5.693.2 Constructor & Destructor Documentation	2843
5.694 std::decimal::decimal32 Class Reference	2844
5.694.1 Detailed Description	2845
5.694.2 Constructor & Destructor Documentation	2845
5.695 std::decimal::decimal64 Class Reference	2845
5.695.1 Detailed Description	2847

5.695.2 Constructor & Destructor Documentation	2847
5.696 std::default_delete< _Tp > Struct Template Reference	2847
5.696.1 Detailed Description	2847
5.696.2 Constructor & Destructor Documentation	2848
5.696.3 Member Function Documentation	2848
5.697 std::default_delete< _Tp[] > Struct Template Reference	2849
5.697.1 Detailed Description	2849
5.697.2 Constructor & Destructor Documentation	2849
5.697.3 Member Function Documentation	2850
5.698 std::defer_lock_t Struct Reference	2850
5.698.1 Detailed Description	2850
5.699 std::deque< _Tp, _Alloc > Class Template Reference	2850
5.699.1 Detailed Description	2855
5.699.2 Constructor & Destructor Documentation	2856
5.699.3 Member Function Documentation	2860
5.699.4 Member Data Documentation	2882
5.700 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference	2882
5.700.1 Detailed Description	2883
5.700.2 Member Typedef Documentation	2883
5.700.3 Constructor & Destructor Documentation	2884
5.700.4 Member Function Documentation	2885
5.700.5 Friends And Related Function Documentation	2888
5.701 std::discrete_distribution< _IntType > Class Template Reference	2889
5.701.1 Detailed Description	2890
5.701.2 Member Typedef Documentation	2891
5.701.3 Member Function Documentation	2891
5.701.4 Friends And Related Function Documentation	2893
5.702 std::discrete_distribution< _IntType >::param_type Struct Reference	2894
5.702.1 Detailed Description	2894
5.703 std::divides< _Tp > Struct Template Reference	2895
5.703.1 Detailed Description	2895
5.703.2 Member Typedef Documentation	2895
5.704 std::divides< void > Struct Template Reference	2896
5.704.1 Detailed Description	2897
5.705 std::domain_error Class Reference	2897
5.705.1 Detailed Description	2897
5.705.2 Member Function Documentation	2898
5.706 std::enable_if< bool, _Tp > Struct Template Reference	2898
5.706.1 Detailed Description	2898



5.707 <code>std::enable_shared_from_this&lt;_Tp&gt;</code> Class Template Reference . . . . .	2898
5.707.1 Detailed Description . . . . .	2899
5.708 <code>std::equal_to&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2899
5.708.1 Detailed Description . . . . .	2900
5.708.2 Member Typedef Documentation . . . . .	2900
5.709 <code>std::equal_to&lt;void&gt;</code> Struct Template Reference . . . . .	2901
5.709.1 Detailed Description . . . . .	2901
5.710 <code>std::error_code</code> Struct Reference . . . . .	2901
5.710.1 Detailed Description . . . . .	2902
5.711 <code>std::error_condition</code> Struct Reference . . . . .	2902
5.711.1 Detailed Description . . . . .	2902
5.712 <code>std::exception</code> Class Reference . . . . .	2903
5.712.1 Detailed Description . . . . .	2903
5.713 <code>std::experimental::filesystem::v1::path</code> Class Reference . . . . .	2904
5.713.1 Detailed Description . . . . .	2906
5.714 <code>std::experimental::filesystem::v1::path::iterator</code> Class Reference . . . . .	2906
5.714.1 Detailed Description . . . . .	2907
5.715 <code>std::experimental::fundamentals_v1::_Has_addressof&lt;_Tp&gt;</code> Struct Template Reference . . . . .	2907
5.715.1 Detailed Description . . . . .	2907
5.716 <code>std::experimental::fundamentals_v1::_Optional_base&lt;_Tp, _ShouldProvideDestructor&gt;</code> Class Template Reference . . . . .	2908
5.716.1 Detailed Description . . . . .	2908
5.717 <code>std::experimental::fundamentals_v1::_Optional_base&lt;_Tp, false&gt;</code> Class Template Reference . . . . .	2909
5.717.1 Detailed Description . . . . .	2909
5.718 <code>std::experimental::fundamentals_v1::any</code> Class Reference . . . . .	2910
5.718.1 Detailed Description . . . . .	2910
5.718.2 Constructor & Destructor Documentation . . . . .	2910
5.718.3 Member Function Documentation . . . . .	2912
5.719 <code>std::experimental::fundamentals_v1::bad_any_cast</code> Class Reference . . . . .	2914
5.719.1 Detailed Description . . . . .	2914
5.719.2 Member Function Documentation . . . . .	2915
5.720 <code>std::experimental::fundamentals_v1::bad_optional_access</code> Class Reference . . . . .	2915
5.720.1 Detailed Description . . . . .	2916
5.720.2 Member Function Documentation . . . . .	2916
5.721 <code>std::experimental::fundamentals_v1::basic_string_view&lt;_CharT, _Traits&gt;</code> Class Template Reference . . . . .	2916
5.721.1 Detailed Description . . . . .	2918
5.722 <code>std::experimental::fundamentals_v1::in_place_t</code> Struct Reference . . . . .	2918
5.722.1 Detailed Description . . . . .	2918
5.723 <code>std::experimental::fundamentals_v1::nullopt_t</code> Struct Reference . . . . .	2919

5.723.1 Detailed Description . . . . .	2919
5.724 std::experimental::fundamentals_v1::optional< _Tp > Class Template Reference . . . . .	2919
5.724.1 Detailed Description . . . . .	2921
5.725 std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits > Class Template Reference . . . . .	2921
5.725.1 Detailed Description . . . . .	2922
5.726 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . . .	2922
5.726.1 Detailed Description . . . . .	2923
5.726.2 Member Typedef Documentation . . . . .	2923
5.727 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . . .	2923
5.727.1 Detailed Description . . . . .	2924
5.727.2 Member Typedef Documentation . . . . .	2924
5.728 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference . . . . .	2925
5.728.1 Detailed Description . . . . .	2926
5.729 std::exponential_distribution< _RealType > Class Template Reference . . . . .	2926
5.729.1 Detailed Description . . . . .	2927
5.729.2 Member Typedef Documentation . . . . .	2927
5.729.3 Constructor & Destructor Documentation . . . . .	2927
5.729.4 Member Function Documentation . . . . .	2928
5.729.5 Friends And Related Function Documentation . . . . .	2929
5.730 std::exponential_distribution< _RealType >::param_type Struct Reference . . . . .	2930
5.730.1 Detailed Description . . . . .	2930
5.731 std::extent< typename, _UInt > Struct Template Reference . . . . .	2931
5.731.1 Detailed Description . . . . .	2931
5.732 std::extreme_value_distribution< _RealType > Class Template Reference . . . . .	2932
5.732.1 Detailed Description . . . . .	2932
5.732.2 Member Typedef Documentation . . . . .	2933
5.732.3 Member Function Documentation . . . . .	2933
5.732.4 Friends And Related Function Documentation . . . . .	2935
5.733 std::extreme_value_distribution< _RealType >::param_type Struct Reference . . . . .	2935
5.733.1 Detailed Description . . . . .	2936
5.734 std::fisher_f_distribution< _RealType > Class Template Reference . . . . .	2936
5.734.1 Detailed Description . . . . .	2937
5.734.2 Member Typedef Documentation . . . . .	2937
5.734.3 Member Function Documentation . . . . .	2937
5.734.4 Friends And Related Function Documentation . . . . .	2939
5.735 std::fisher_f_distribution< _RealType >::param_type Struct Reference . . . . .	2940
5.735.1 Detailed Description . . . . .	2941
5.736 std::forward_iterator_tag Struct Reference . . . . .	2941

5.736.1 Detailed Description . . . . .	2942
5.737 std::forward_list< _Tp, _Alloc > Class Template Reference . . . . .	2942
5.737.1 Detailed Description . . . . .	2944
5.737.2 Constructor & Destructor Documentation . . . . .	2945
5.737.3 Member Function Documentation . . . . .	2949
5.738 std::fpos< _StateT > Class Template Reference . . . . .	2967
5.738.1 Detailed Description . . . . .	2967
5.738.2 Constructor & Destructor Documentation . . . . .	2968
5.738.3 Member Function Documentation . . . . .	2968
5.739 std::front_insert_iterator< _Container > Class Template Reference . . . . .	2970
5.739.1 Detailed Description . . . . .	2971
5.739.2 Member Typedef Documentation . . . . .	2971
5.739.3 Constructor & Destructor Documentation . . . . .	2973
5.739.4 Member Function Documentation . . . . .	2973
5.740 std::function< _Res(_ArgTypes...)> Class Template Reference . . . . .	2974
5.740.1 Detailed Description . . . . .	2976
5.740.2 Constructor & Destructor Documentation . . . . .	2976
5.740.3 Member Function Documentation . . . . .	2978
5.741 std::future< _Res > Class Template Reference . . . . .	2983
5.741.1 Detailed Description . . . . .	2985
5.741.2 Member Typedef Documentation . . . . .	2985
5.741.3 Constructor & Destructor Documentation . . . . .	2985
5.741.4 Member Function Documentation . . . . .	2985
5.742 std::future< _Res & > Class Template Reference . . . . .	2986
5.742.1 Detailed Description . . . . .	2987
5.742.2 Member Typedef Documentation . . . . .	2987
5.742.3 Constructor & Destructor Documentation . . . . .	2988
5.742.4 Member Function Documentation . . . . .	2988
5.743 std::future< void > Class Template Reference . . . . .	2989
5.743.1 Detailed Description . . . . .	2990
5.743.2 Member Typedef Documentation . . . . .	2990
5.743.3 Constructor & Destructor Documentation . . . . .	2991
5.743.4 Member Function Documentation . . . . .	2991
5.744 std::future_error Class Reference . . . . .	2992
5.744.1 Detailed Description . . . . .	2992
5.744.2 Member Function Documentation . . . . .	2992
5.745 std::gamma_distribution< _RealType > Class Template Reference . . . . .	2993
5.745.1 Detailed Description . . . . .	2994
5.745.2 Member Typedef Documentation . . . . .	2994

5.745.3 Constructor & Destructor Documentation	2994
5.745.4 Member Function Documentation	2995
5.745.5 Friends And Related Function Documentation	2997
5.746 std::gamma_distribution<_RealType>::param_type Struct Reference	2998
5.746.1 Detailed Description	2999
5.747 std::geometric_distribution<_IntType> Class Template Reference	2999
5.747.1 Detailed Description	3000
5.747.2 Member Typedef Documentation	3000
5.747.3 Member Function Documentation	3001
5.747.4 Friends And Related Function Documentation	3002
5.748 std::geometric_distribution<_IntType>::param_type Struct Reference	3003
5.748.1 Detailed Description	3003
5.749 std::greater<_Tp> Struct Template Reference	3004
5.749.1 Detailed Description	3004
5.749.2 Member Typedef Documentation	3004
5.750 std::greater<void> Struct Template Reference	3005
5.750.1 Detailed Description	3006
5.751 std::greater_equal<_Tp> Struct Template Reference	3006
5.751.1 Detailed Description	3007
5.751.2 Member Typedef Documentation	3007
5.752 std::greater_equal<void> Struct Template Reference	3008
5.752.1 Detailed Description	3008
5.753 std::gslice Class Reference	3008
5.753.1 Detailed Description	3009
5.754 std::gslice_array<_Tp> Class Template Reference	3009
5.754.1 Detailed Description	3010
5.755 std::has_virtual_destructor<_Tp> Struct Template Reference	3011
5.755.1 Detailed Description	3011
5.756 std::hash<_Tp> Struct Template Reference	3012
5.756.1 Detailed Description	3012
5.757 std::hash<__debug::bitset<_Nb>> Struct Template Reference	3012
5.757.1 Detailed Description	3012
5.758 std::hash<__debug::vector<bool,_Alloc>> Struct Template Reference	3013
5.758.1 Detailed Description	3013
5.759 std::hash<__gnu_cxx::__u16vstring> Struct Template Reference	3013
5.759.1 Detailed Description	3014
5.760 std::hash<__gnu_cxx::__u32vstring> Struct Template Reference	3014
5.760.1 Detailed Description	3014
5.761 std::hash<__gnu_cxx::__vstring> Struct Template Reference	3014

5.761.1 Detailed Description . . . . .	3015
5.762 std::hash< __gnu_cxx::__wvstring > Struct Template Reference . . . . .	3015
5.762.1 Detailed Description . . . . .	3015
5.763 std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference . . . . .	3016
5.763.1 Detailed Description . . . . .	3016
5.763.2 Member Typedef Documentation . . . . .	3016
5.764 std::hash< __gnu_cxx::throw_value_random > Struct Template Reference . . . . .	3017
5.764.1 Detailed Description . . . . .	3018
5.764.2 Member Typedef Documentation . . . . .	3018
5.765 std::hash< __profile::bitset< _Nb > > Struct Template Reference . . . . .	3018
5.765.1 Detailed Description . . . . .	3019
5.766 std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference . . . . .	3019
5.766.1 Detailed Description . . . . .	3019
5.767 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference . . . . .	3020
5.767.1 Detailed Description . . . . .	3020
5.768 std::hash< _Tp * > Struct Template Reference . . . . .	3020
5.768.1 Detailed Description . . . . .	3021
5.769 std::hash< bool > Struct Template Reference . . . . .	3021
5.769.1 Detailed Description . . . . .	3021
5.770 std::hash< char > Struct Template Reference . . . . .	3021
5.770.1 Detailed Description . . . . .	3022
5.771 std::hash< char16_t > Struct Template Reference . . . . .	3022
5.771.1 Detailed Description . . . . .	3022
5.772 std::hash< char32_t > Struct Template Reference . . . . .	3023
5.772.1 Detailed Description . . . . .	3023
5.773 std::hash< double > Struct Template Reference . . . . .	3023
5.773.1 Detailed Description . . . . .	3024
5.774 std::hash< error_code > Struct Template Reference . . . . .	3024
5.774.1 Detailed Description . . . . .	3024
5.775 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference . . . . .	3024
5.775.1 Detailed Description . . . . .	3025
5.776 std::hash< float > Struct Template Reference . . . . .	3025
5.776.1 Detailed Description . . . . .	3025
5.777 std::hash< int > Struct Template Reference . . . . .	3026
5.777.1 Detailed Description . . . . .	3026
5.778 std::hash< long > Struct Template Reference . . . . .	3026
5.778.1 Detailed Description . . . . .	3027
5.779 std::hash< long double > Struct Template Reference . . . . .	3027
5.779.1 Detailed Description . . . . .	3027

5.780 <code>std::hash&lt; long long &gt;</code> Struct Template Reference . . . . .	3027
5.780.1 Detailed Description . . . . .	3028
5.781 <code>std::hash&lt; shared_ptr&lt; _Tp &gt; &gt;</code> Struct Template Reference . . . . .	3028
5.781.1 Detailed Description . . . . .	3028
5.782 <code>std::hash&lt; short &gt;</code> Struct Template Reference . . . . .	3029
5.782.1 Detailed Description . . . . .	3029
5.783 <code>std::hash&lt; signed char &gt;</code> Struct Template Reference . . . . .	3029
5.783.1 Detailed Description . . . . .	3030
5.784 <code>std::hash&lt; string &gt;</code> Struct Template Reference . . . . .	3030
5.784.1 Detailed Description . . . . .	3030
5.785 <code>std::hash&lt; thread::id &gt;</code> Struct Template Reference . . . . .	3030
5.785.1 Detailed Description . . . . .	3031
5.786 <code>std::hash&lt; type_index &gt;</code> Struct Template Reference . . . . .	3031
5.786.1 Detailed Description . . . . .	3031
5.787 <code>std::hash&lt; u16string &gt;</code> Struct Template Reference . . . . .	3032
5.787.1 Detailed Description . . . . .	3032
5.788 <code>std::hash&lt; u32string &gt;</code> Struct Template Reference . . . . .	3032
5.788.1 Detailed Description . . . . .	3033
5.789 <code>std::hash&lt; unique_ptr&lt; _Tp, _Dp &gt; &gt;</code> Struct Template Reference . . . . .	3033
5.789.1 Detailed Description . . . . .	3033
5.790 <code>std::hash&lt; unsigned char &gt;</code> Struct Template Reference . . . . .	3034
5.790.1 Detailed Description . . . . .	3034
5.791 <code>std::hash&lt; unsigned int &gt;</code> Struct Template Reference . . . . .	3034
5.791.1 Detailed Description . . . . .	3035
5.792 <code>std::hash&lt; unsigned long &gt;</code> Struct Template Reference . . . . .	3035
5.792.1 Detailed Description . . . . .	3035
5.793 <code>std::hash&lt; unsigned long long &gt;</code> Struct Template Reference . . . . .	3035
5.793.1 Detailed Description . . . . .	3036
5.794 <code>std::hash&lt; unsigned short &gt;</code> Struct Template Reference . . . . .	3036
5.794.1 Detailed Description . . . . .	3036
5.795 <code>std::hash&lt; wchar_t &gt;</code> Struct Template Reference . . . . .	3037
5.795.1 Detailed Description . . . . .	3037
5.796 <code>std::hash&lt; wstring &gt;</code> Struct Template Reference . . . . .	3037
5.796.1 Detailed Description . . . . .	3038
5.797 <code>std::hash&lt;::bitset&lt; _Nb &gt; &gt;</code> Struct Template Reference . . . . .	3038
5.797.1 Detailed Description . . . . .	3038
5.798 <code>std::hash&lt;::vector&lt; bool, _Alloc &gt; &gt;</code> Struct Template Reference . . . . .	3038
5.798.1 Detailed Description . . . . .	3039
5.799 <code>std::independent_bits_engine&lt; _RandomNumberEngine, __w, _UIntType &gt;</code> Class Template Reference	3039

5.799.1 Detailed Description . . . . .	3040
5.799.2 Member Typedef Documentation . . . . .	3040
5.799.3 Constructor & Destructor Documentation . . . . .	3040
5.799.4 Member Function Documentation . . . . .	3042
5.799.5 Friends And Related Function Documentation . . . . .	3044
5.800 std::indirect_array< _Tp > Class Template Reference . . . . .	3045
5.800.1 Detailed Description . . . . .	3047
5.801 std::initializer_list< _E > Class Template Reference . . . . .	3047
5.801.1 Detailed Description . . . . .	3048
5.802 std::input_iterator_tag Struct Reference . . . . .	3048
5.802.1 Detailed Description . . . . .	3048
5.803 std::insert_iterator< _Container > Class Template Reference . . . . .	3049
5.803.1 Detailed Description . . . . .	3050
5.803.2 Member Typedef Documentation . . . . .	3050
5.803.3 Constructor & Destructor Documentation . . . . .	3051
5.803.4 Member Function Documentation . . . . .	3051
5.804 std::integer_sequence< _Tp, _Idx > Struct Template Reference . . . . .	3053
5.804.1 Detailed Description . . . . .	3053
5.805 std::integral_constant< _Tp, __v > Struct Template Reference . . . . .	3054
5.805.1 Detailed Description . . . . .	3055
5.806 std::invalid_argument Class Reference . . . . .	3055
5.806.1 Detailed Description . . . . .	3056
5.806.2 Member Function Documentation . . . . .	3056
5.807 std::ios_base Class Reference . . . . .	3056
5.807.1 Detailed Description . . . . .	3059
5.807.2 Member Typedef Documentation . . . . .	3059
5.807.3 Member Enumeration Documentation . . . . .	3061
5.807.4 Constructor & Destructor Documentation . . . . .	3062
5.807.5 Member Function Documentation . . . . .	3062
5.807.6 Member Data Documentation . . . . .	3069
5.808 std::ios_base::failure Class Reference . . . . .	3076
5.808.1 Detailed Description . . . . .	3076
5.808.2 Member Function Documentation . . . . .	3077
5.809 std::is_abstract< _Tp > Struct Template Reference . . . . .	3077
5.809.1 Detailed Description . . . . .	3078
5.810 std::is_arithmetic< _Tp > Struct Template Reference . . . . .	3078
5.810.1 Detailed Description . . . . .	3078
5.811 std::is_array< typename > Struct Template Reference . . . . .	3078
5.811.1 Detailed Description . . . . .	3079

5.812 std::is_assignable< _Tp, _Up > Struct Template Reference . . . . .	3079
5.812.1 Detailed Description . . . . .	3080
5.813 std::is_base_of< _Base, _Derived > Struct Template Reference . . . . .	3080
5.813.1 Detailed Description . . . . .	3081
5.814 std::is_bind_expression< _Tp > Struct Template Reference . . . . .	3081
5.814.1 Detailed Description . . . . .	3082
5.815 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference . . . . .	3082
5.815.1 Detailed Description . . . . .	3083
5.816 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	3083
5.816.1 Detailed Description . . . . .	3084
5.817 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference . . . . .	3084
5.817.1 Detailed Description . . . . .	3085
5.818 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	3085
5.818.1 Detailed Description . . . . .	3086
5.819 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference . . . . .	3086
5.819.1 Detailed Description . . . . .	3087
5.820 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	3087
5.820.1 Detailed Description . . . . .	3088
5.821 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference . . . . .	3088
5.821.1 Detailed Description . . . . .	3089
5.822 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference . . . . .	3089
5.822.1 Detailed Description . . . . .	3090
5.823 std::is_class< _Tp > Struct Template Reference . . . . .	3090
5.823.1 Detailed Description . . . . .	3091
5.824 std::is_compound< _Tp > Struct Template Reference . . . . .	3091
5.824.1 Detailed Description . . . . .	3092
5.825 std::is_const< typename > Struct Template Reference . . . . .	3092
5.825.1 Detailed Description . . . . .	3093
5.826 std::is_constructible< _Tp, _Args > Struct Template Reference . . . . .	3093
5.826.1 Detailed Description . . . . .	3094
5.827 std::is_convertible< _From, _To > Struct Template Reference . . . . .	3094
5.827.1 Detailed Description . . . . .	3094
5.828 std::is_copy_assignable< _Tp > Struct Template Reference . . . . .	3095
5.828.1 Detailed Description . . . . .	3095
5.829 std::is_copy_constructible< _Tp > Struct Template Reference . . . . .	3095
5.829.1 Detailed Description . . . . .	3095
5.830 std::is_default_constructible< _Tp > Struct Template Reference . . . . .	3096
5.830.1 Detailed Description . . . . .	3096



5.831 <code>std::is_destructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3097
5.831.1 Detailed Description . . . . .	3097
5.832 <code>std::is_empty&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3097
5.832.1 Detailed Description . . . . .	3098
5.833 <code>std::is_enum&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3098
5.833.1 Detailed Description . . . . .	3099
5.834 <code>std::is_error_code_enum&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3099
5.834.1 Detailed Description . . . . .	3100
5.835 <code>std::is_error_code_enum&lt; future_errc &gt;</code> Struct Template Reference . . . . .	3100
5.835.1 Detailed Description . . . . .	3101
5.836 <code>std::is_error_condition_enum&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3101
5.836.1 Detailed Description . . . . .	3102
5.837 <code>std::is_final&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3102
5.837.1 Detailed Description . . . . .	3103
5.838 <code>std::is_floating_point&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3103
5.838.1 Detailed Description . . . . .	3103
5.839 <code>std::is_function&lt; typename &gt;</code> Struct Template Reference . . . . .	3103
5.839.1 Detailed Description . . . . .	3104
5.840 <code>std::is_fundamental&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3104
5.840.1 Detailed Description . . . . .	3104
5.841 <code>std::is_integral&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3105
5.841.1 Detailed Description . . . . .	3105
5.842 <code>std::is_literal_type&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3106
5.842.1 Detailed Description . . . . .	3106
5.843 <code>std::is_lvalue_reference&lt; typename &gt;</code> Struct Template Reference . . . . .	3107
5.843.1 Detailed Description . . . . .	3107
5.844 <code>std::is_member_function_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3108
5.844.1 Detailed Description . . . . .	3108
5.845 <code>std::is_member_object_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3109
5.845.1 Detailed Description . . . . .	3109
5.846 <code>std::is_member_pointer&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3109
5.846.1 Detailed Description . . . . .	3110
5.847 <code>std::is_move_assignable&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3110
5.847.1 Detailed Description . . . . .	3110
5.848 <code>std::is_move_constructible&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3110
5.848.1 Detailed Description . . . . .	3111
5.849 <code>std::is_nothrow_assignable&lt; _Tp, _Up &gt;</code> Struct Template Reference . . . . .	3111
5.849.1 Detailed Description . . . . .	3111
5.850 <code>std::is_nothrow_constructible&lt; _Tp, _Args &gt;</code> Struct Template Reference . . . . .	3111

---

5.850.1 Detailed Description	3112
5.851 <a href="#">std::is_nothrow_copy_assignable&lt; _Tp &gt; Struct Template Reference</a>	3112
5.851.1 Detailed Description	3112
5.852 <a href="#">std::is_nothrow_copy_constructible&lt; _Tp &gt; Struct Template Reference</a>	3113
5.852.1 Detailed Description	3113
5.853 <a href="#">std::is_nothrow_default_constructible&lt; _Tp &gt; Struct Template Reference</a>	3113
5.853.1 Detailed Description	3114
5.854 <a href="#">std::is_nothrow_destructible&lt; _Tp &gt; Struct Template Reference</a>	3114
5.854.1 Detailed Description	3114
5.855 <a href="#">std::is_nothrow_move_assignable&lt; _Tp &gt; Struct Template Reference</a>	3114
5.855.1 Detailed Description	3115
5.856 <a href="#">std::is_nothrow_move_constructible&lt; _Tp &gt; Struct Template Reference</a>	3115
5.856.1 Detailed Description	3115
5.857 <a href="#">std::is_nothrow_swappable&lt; _Tp &gt; Struct Template Reference</a>	3115
5.857.1 Detailed Description	3115
5.858 <a href="#">std::is_nothrow_swappable_with&lt; _Tp, _Up &gt; Struct Template Reference</a>	3116
5.858.1 Detailed Description	3116
5.859 <a href="#">std::is_null_pointer&lt; _Tp &gt; Struct Template Reference</a>	3116
5.859.1 Detailed Description	3116
5.860 <a href="#">std::is_object&lt; _Tp &gt; Struct Template Reference</a>	3117
5.860.1 Detailed Description	3117
5.861 <a href="#">std::is_placeholder&lt; _Tp &gt; Struct Template Reference</a>	3118
5.861.1 Detailed Description	3118
5.862 <a href="#">std::is_placeholder&lt; _Placeholder&lt; _Num &gt; &gt; Struct Template Reference</a>	3119
5.862.1 Detailed Description	3119
5.863 <a href="#">std::is_pod&lt; _Tp &gt; Struct Template Reference</a>	3120
5.863.1 Detailed Description	3120
5.864 <a href="#">std::is_pointer&lt; _Tp &gt; Struct Template Reference</a>	3121
5.864.1 Detailed Description	3121
5.865 <a href="#">std::is_polymorphic&lt; _Tp &gt; Struct Template Reference</a>	3121
5.865.1 Detailed Description	3122
5.866 <a href="#">std::is_reference&lt; _Tp &gt; Struct Template Reference</a>	3122
5.866.1 Detailed Description	3122
5.867 <a href="#">std::is_rvalue_reference&lt; typename &gt; Struct Template Reference</a>	3122
5.867.1 Detailed Description	3123
5.868 <a href="#">std::is_same&lt; typename, typename &gt; Struct Template Reference</a>	3123
5.868.1 Detailed Description	3124
5.869 <a href="#">std::is_scalar&lt; _Tp &gt; Struct Template Reference</a>	3124
5.869.1 Detailed Description	3124

5.870 <a href="#">std::is_standard_layout&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3125
5.870.1 Detailed Description . . . . .	3125
5.871 <a href="#">std::is_swappable&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3126
5.871.1 Detailed Description . . . . .	3126
5.872 <a href="#">std::is_swappable_with&lt; _Tp, _Up &gt; Struct Template Reference</a> . . . . .	3126
5.872.1 Detailed Description . . . . .	3126
5.873 <a href="#">std::is_trivial&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3127
5.873.1 Detailed Description . . . . .	3127
5.874 <a href="#">std::is_trivially_assignable&lt; _Tp, _Up &gt; Struct Template Reference</a> . . . . .	3128
5.874.1 Detailed Description . . . . .	3128
5.875 <a href="#">std::is_trivially_constructible&lt; _Tp, _Args &gt; Struct Template Reference</a> . . . . .	3129
5.875.1 Detailed Description . . . . .	3129
5.876 <a href="#">std::is_trivially_default_constructible&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3129
5.876.1 Detailed Description . . . . .	3129
5.877 <a href="#">std::is_trivially_destructible&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3129
5.877.1 Detailed Description . . . . .	3130
5.878 <a href="#">std::is_union&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3130
5.878.1 Detailed Description . . . . .	3131
5.879 <a href="#">std::is_void&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3131
5.879.1 Detailed Description . . . . .	3132
5.880 <a href="#">std::is_volatile&lt; typename &gt; Struct Template Reference</a> . . . . .	3132
5.880.1 Detailed Description . . . . .	3133
5.881 <a href="#">std::istream_iterator&lt; _Tp, _CharT, _Traits, _Dist &gt; Class Template Reference</a> . . . . .	3133
5.881.1 Detailed Description . . . . .	3134
5.881.2 Member Typedef Documentation . . . . .	3134
5.881.3 Constructor & Destructor Documentation . . . . .	3135
5.882 <a href="#">std::istreambuf_iterator&lt; _CharT, _Traits &gt; Class Template Reference</a> . . . . .	3136
5.882.1 Detailed Description . . . . .	3137
5.882.2 Member Typedef Documentation . . . . .	3137
5.882.3 Constructor & Destructor Documentation . . . . .	3139
5.882.4 Member Function Documentation . . . . .	3140
5.883 <a href="#">std::iterator&lt; _Category, _Tp, _Distance, _Pointer, _Reference &gt; Struct Template Reference</a> . . . . .	3141
5.883.1 Detailed Description . . . . .	3142
5.883.2 Member Typedef Documentation . . . . .	3142
5.884 <a href="#">std::iterator_traits&lt; _Tp * &gt; Struct Template Reference</a> . . . . .	3143
5.884.1 Detailed Description . . . . .	3144
5.885 <a href="#">std::iterator_traits&lt; const _Tp * &gt; Struct Template Reference</a> . . . . .	3144
5.885.1 Detailed Description . . . . .	3144
5.886 <a href="#">std::length_error Class Reference</a> . . . . .	3145

5.886.1 Detailed Description	3145
5.886.2 Member Function Documentation	3145
5.887 std::less< _Tp > Struct Template Reference	3146
5.887.1 Detailed Description	3146
5.887.2 Member Typedef Documentation	3146
5.888 std::less< void > Struct Template Reference	3147
5.888.1 Detailed Description	3148
5.889 std::less_equal< _Tp > Struct Template Reference	3148
5.889.1 Detailed Description	3148
5.889.2 Member Typedef Documentation	3149
5.890 std::less_equal< void > Struct Template Reference	3149
5.890.1 Detailed Description	3150
5.891 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	3150
5.891.1 Detailed Description	3151
5.891.2 Member Typedef Documentation	3151
5.891.3 Constructor & Destructor Documentation	3151
5.891.4 Member Function Documentation	3152
5.891.5 Friends And Related Function Documentation	3154
5.891.6 Member Data Documentation	3156
5.892 std::list< _Tp, _Alloc > Class Template Reference	3157
5.892.1 Detailed Description	3160
5.892.2 Constructor & Destructor Documentation	3161
5.892.3 Member Function Documentation	3164
5.893 std::locale Class Reference	3185
5.893.1 Detailed Description	3187
5.893.2 Member Typedef Documentation	3187
5.893.3 Constructor & Destructor Documentation	3187
5.893.4 Member Function Documentation	3191
5.893.5 Friends And Related Function Documentation	3194
5.893.6 Member Data Documentation	3195
5.894 std::locale::facet Class Reference	3198
5.894.1 Detailed Description	3199
5.894.2 Constructor & Destructor Documentation	3199
5.895 std::locale::id Class Reference	3200
5.895.1 Detailed Description	3200
5.895.2 Constructor & Destructor Documentation	3200
5.895.3 Friends And Related Function Documentation	3201
5.896 std::lock_guard< _Mutex > Class Template Reference	3202
5.896.1 Detailed Description	3202

5.897 std::logic_error Class Reference . . . . .	3203
5.897.1 Detailed Description . . . . .	3203
5.897.2 Constructor & Destructor Documentation . . . . .	3203
5.897.3 Member Function Documentation . . . . .	3204
5.898 std::logical_and< _Tp > Struct Template Reference . . . . .	3204
5.898.1 Detailed Description . . . . .	3205
5.898.2 Member Typedef Documentation . . . . .	3205
5.899 std::logical_and< void > Struct Template Reference . . . . .	3206
5.899.1 Detailed Description . . . . .	3206
5.900 std::logical_not< _Tp > Struct Template Reference . . . . .	3206
5.900.1 Detailed Description . . . . .	3207
5.900.2 Member Typedef Documentation . . . . .	3207
5.901 std::logical_not< void > Struct Template Reference . . . . .	3208
5.901.1 Detailed Description . . . . .	3208
5.902 std::logical_or< _Tp > Struct Template Reference . . . . .	3208
5.902.1 Detailed Description . . . . .	3209
5.902.2 Member Typedef Documentation . . . . .	3209
5.903 std::logical_or< void > Struct Template Reference . . . . .	3210
5.903.1 Detailed Description . . . . .	3210
5.904 std::lognormal_distribution< _RealType > Class Template Reference . . . . .	3210
5.904.1 Detailed Description . . . . .	3211
5.904.2 Member Typedef Documentation . . . . .	3212
5.904.3 Member Function Documentation . . . . .	3212
5.904.4 Friends And Related Function Documentation . . . . .	3213
5.905 std::lognormal_distribution< _RealType >::param_type Struct Reference . . . . .	3215
5.905.1 Detailed Description . . . . .	3215
5.906 std::make_signed< _Tp > Struct Template Reference . . . . .	3215
5.906.1 Detailed Description . . . . .	3215
5.907 std::make_unsigned< _Tp > Struct Template Reference . . . . .	3216
5.907.1 Detailed Description . . . . .	3216
5.908 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . . .	3216
5.908.1 Detailed Description . . . . .	3219
5.908.2 Constructor & Destructor Documentation . . . . .	3220
5.908.3 Member Function Documentation . . . . .	3224
5.909 std::mask_array< _Tp > Class Template Reference . . . . .	3250
5.909.1 Detailed Description . . . . .	3251
5.910 std::match_results< _Bi_iter, _Alloc > Class Template Reference . . . . .	3252
5.910.1 Detailed Description . . . . .	3256
5.910.2 Constructor & Destructor Documentation . . . . .	3256

5.910.3 Member Function Documentation . . . . .	3257
5.911 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	3265
5.911.1 Detailed Description . . . . .	3265
5.911.2 Member Typedef Documentation . . . . .	3265
5.912 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . . .	3266
5.912.1 Detailed Description . . . . .	3267
5.912.2 Member Typedef Documentation . . . . .	3267
5.913 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . . .	3268
5.913.1 Detailed Description . . . . .	3268
5.913.2 Member Typedef Documentation . . . . .	3269
5.914 std::mem_fun_t< _Ret, _Tp > Class Template Reference . . . . .	3269
5.914.1 Detailed Description . . . . .	3270
5.914.2 Member Typedef Documentation . . . . .	3270
5.915 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference . . . . .	3270
5.915.1 Detailed Description . . . . .	3272
5.915.2 Member Typedef Documentation . . . . .	3272
5.915.3 Constructor & Destructor Documentation . . . . .	3273
5.915.4 Member Function Documentation . . . . .	3273
5.915.5 Friends And Related Function Documentation . . . . .	3274
5.916 std::messages< _CharT > Class Template Reference . . . . .	3276
5.916.1 Detailed Description . . . . .	3278
5.916.2 Member Typedef Documentation . . . . .	3278
5.916.3 Constructor & Destructor Documentation . . . . .	3278
5.916.4 Member Function Documentation . . . . .	3279
5.916.5 Member Data Documentation . . . . .	3280
5.917 std::messages_base Struct Reference . . . . .	3280
5.917.1 Detailed Description . . . . .	3281
5.918 std::messages_byname< _CharT > Class Template Reference . . . . .	3281
5.918.1 Detailed Description . . . . .	3283
5.918.2 Member Function Documentation . . . . .	3283
5.918.3 Member Data Documentation . . . . .	3283
5.919 std::minus< _Tp > Struct Template Reference . . . . .	3284
5.919.1 Detailed Description . . . . .	3284
5.919.2 Member Typedef Documentation . . . . .	3284
5.920 std::minus< void > Struct Template Reference . . . . .	3285
5.920.1 Detailed Description . . . . .	3286
5.921 std::modulus< _Tp > Struct Template Reference . . . . .	3286
5.921.1 Detailed Description . . . . .	3286

5.921.2 Member Typedef Documentation . . . . .	3287
5.922 std::modulus< void > Struct Template Reference . . . . .	3287
5.922.1 Detailed Description . . . . .	3288
5.923 std::money_base Class Reference . . . . .	3288
5.923.1 Detailed Description . . . . .	3289
5.924 std::money_get< _CharT, _InIter > Class Template Reference . . . . .	3289
5.924.1 Detailed Description . . . . .	3290
5.924.2 Member Typedef Documentation . . . . .	3291
5.924.3 Constructor & Destructor Documentation . . . . .	3291
5.924.4 Member Function Documentation . . . . .	3292
5.924.5 Member Data Documentation . . . . .	3295
5.925 std::money_put< _CharT, _OutIter > Class Template Reference . . . . .	3295
5.925.1 Detailed Description . . . . .	3296
5.925.2 Member Typedef Documentation . . . . .	3297
5.925.3 Constructor & Destructor Documentation . . . . .	3297
5.925.4 Member Function Documentation . . . . .	3298
5.925.5 Member Data Documentation . . . . .	3301
5.926 std::moneypunct< _CharT, _Intl > Class Template Reference . . . . .	3302
5.926.1 Detailed Description . . . . .	3303
5.926.2 Member Typedef Documentation . . . . .	3304
5.926.3 Constructor & Destructor Documentation . . . . .	3304
5.926.4 Member Function Documentation . . . . .	3306
5.926.5 Member Data Documentation . . . . .	3314
5.927 std::moneypunct_byname< _CharT, _Intl > Class Template Reference . . . . .	3315
5.927.1 Detailed Description . . . . .	3317
5.927.2 Member Function Documentation . . . . .	3317
5.927.3 Member Data Documentation . . . . .	3326
5.928 std::move_iterator< _Iterator > Class Template Reference . . . . .	3326
5.928.1 Detailed Description . . . . .	3327
5.929 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . . .	3328
5.929.1 Detailed Description . . . . .	3331
5.929.2 Constructor & Destructor Documentation . . . . .	3331
5.929.3 Member Function Documentation . . . . .	3335
5.930 std::multiplies< _Tp > Struct Template Reference . . . . .	3359
5.930.1 Detailed Description . . . . .	3359
5.930.2 Member Typedef Documentation . . . . .	3359
5.931 std::multiplies< void > Struct Template Reference . . . . .	3360
5.931.1 Detailed Description . . . . .	3361
5.932 std::multiset< _Key, _Compare, _Alloc > Class Template Reference . . . . .	3361

5.932.1 Detailed Description	3363
5.932.2 Constructor & Destructor Documentation	3364
5.932.3 Member Function Documentation	3368
5.933 std::mutex Class Reference	3387
5.933.1 Detailed Description	3387
5.934 std::negate< _Tp > Struct Template Reference	3388
5.934.1 Detailed Description	3388
5.934.2 Member Typedef Documentation	3388
5.935 std::negate< void > Struct Template Reference	3389
5.935.1 Detailed Description	3389
5.936 std::negative_binomial_distribution< _IntType > Class Template Reference	3389
5.936.1 Detailed Description	3390
5.936.2 Member Typedef Documentation	3391
5.936.3 Member Function Documentation	3391
5.936.4 Friends And Related Function Documentation	3393
5.937 std::negative_binomial_distribution< _IntType >::param_type Struct Reference	3394
5.937.1 Detailed Description	3395
5.938 std::nested_exception Class Reference	3395
5.938.1 Detailed Description	3395
5.939 std::normal_distribution< _RealType > Class Template Reference	3395
5.939.1 Detailed Description	3396
5.939.2 Member Typedef Documentation	3397
5.939.3 Constructor & Destructor Documentation	3397
5.939.4 Member Function Documentation	3397
5.939.5 Friends And Related Function Documentation	3400
5.940 std::normal_distribution< _RealType >::param_type Struct Reference	3401
5.940.1 Detailed Description	3401
5.941 std::not_equal_to< _Tp > Struct Template Reference	3402
5.941.1 Detailed Description	3402
5.941.2 Member Typedef Documentation	3402
5.942 std::not_equal_to< void > Struct Template Reference	3403
5.942.1 Detailed Description	3404
5.943 std::num_get< _CharT, _InIter > Class Template Reference	3404
5.943.1 Detailed Description	3406
5.943.2 Member Typedef Documentation	3406
5.943.3 Constructor & Destructor Documentation	3407
5.943.4 Member Function Documentation	3408
5.943.5 Member Data Documentation	3425
5.944 std::num_put< _CharT, _OutIter > Class Template Reference	3425



5.944.1 Detailed Description . . . . .	3427
5.944.2 Member Typedef Documentation . . . . .	3427
5.944.3 Constructor & Destructor Documentation . . . . .	3428
5.944.4 Member Function Documentation . . . . .	3429
5.944.5 Member Data Documentation . . . . .	3442
5.945 std::numeric_limits< _Tp > Struct Template Reference . . . . .	3442
5.945.1 Detailed Description . . . . .	3443
5.945.2 Member Function Documentation . . . . .	3444
5.945.3 Member Data Documentation . . . . .	3446
5.946 std::numeric_limits< bool > Struct Template Reference . . . . .	3451
5.946.1 Detailed Description . . . . .	3451
5.947 std::numeric_limits< char > Struct Template Reference . . . . .	3452
5.947.1 Detailed Description . . . . .	3452
5.948 std::numeric_limits< char16_t > Struct Template Reference . . . . .	3453
5.948.1 Detailed Description . . . . .	3453
5.949 std::numeric_limits< char32_t > Struct Template Reference . . . . .	3454
5.949.1 Detailed Description . . . . .	3454
5.950 std::numeric_limits< double > Struct Template Reference . . . . .	3455
5.950.1 Detailed Description . . . . .	3455
5.951 std::numeric_limits< float > Struct Template Reference . . . . .	3456
5.951.1 Detailed Description . . . . .	3456
5.952 std::numeric_limits< int > Struct Template Reference . . . . .	3457
5.952.1 Detailed Description . . . . .	3457
5.953 std::numeric_limits< long > Struct Template Reference . . . . .	3458
5.953.1 Detailed Description . . . . .	3458
5.954 std::numeric_limits< long double > Struct Template Reference . . . . .	3459
5.954.1 Detailed Description . . . . .	3459
5.955 std::numeric_limits< long long > Struct Template Reference . . . . .	3460
5.955.1 Detailed Description . . . . .	3460
5.956 std::numeric_limits< short > Struct Template Reference . . . . .	3461
5.956.1 Detailed Description . . . . .	3461
5.957 std::numeric_limits< signed char > Struct Template Reference . . . . .	3462
5.957.1 Detailed Description . . . . .	3462
5.958 std::numeric_limits< unsigned char > Struct Template Reference . . . . .	3463
5.958.1 Detailed Description . . . . .	3463
5.959 std::numeric_limits< unsigned int > Struct Template Reference . . . . .	3464
5.959.1 Detailed Description . . . . .	3464
5.960 std::numeric_limits< unsigned long > Struct Template Reference . . . . .	3465
5.960.1 Detailed Description . . . . .	3465

5.961 <a href="#">std::numeric_limits&lt; unsigned long long &gt; Struct Template Reference</a>	3466
5.961.1 Detailed Description	3466
5.962 <a href="#">std::numeric_limits&lt; unsigned short &gt; Struct Template Reference</a>	3467
5.962.1 Detailed Description	3467
5.963 <a href="#">std::numeric_limits&lt; wchar_t &gt; Struct Template Reference</a>	3468
5.963.1 Detailed Description	3468
5.964 <a href="#">std::numprint&lt; _CharT &gt; Class Template Reference</a>	3469
5.964.1 Detailed Description	3470
5.964.2 Member Typedef Documentation	3470
5.964.3 Constructor & Destructor Documentation	3471
5.964.4 Member Function Documentation	3472
5.964.5 Member Data Documentation	3477
5.965 <a href="#">std::numprint_byname&lt; _CharT &gt; Class Template Reference</a>	3477
5.965.1 Detailed Description	3479
5.965.2 Member Function Documentation	3479
5.965.3 Member Data Documentation	3483
5.966 <a href="#">std::once_flag Struct Reference</a>	3483
5.966.1 Detailed Description	3484
5.966.2 Constructor & Destructor Documentation	3484
5.966.3 Member Function Documentation	3484
5.966.4 Friends And Related Function Documentation	3484
5.967 <a href="#">std::ostream_iterator&lt; _Tp, _CharT, _Traits &gt; Class Template Reference</a>	3485
5.967.1 Detailed Description	3486
5.967.2 Member Typedef Documentation	3486
5.967.3 Constructor & Destructor Documentation	3488
5.967.4 Member Function Documentation	3489
5.968 <a href="#">std::ostreambuf_iterator&lt; _CharT, _Traits &gt; Class Template Reference</a>	3490
5.968.1 Detailed Description	3491
5.968.2 Member Typedef Documentation	3491
5.968.3 Constructor & Destructor Documentation	3493
5.968.4 Member Function Documentation	3493
5.969 <a href="#">std::out_of_range Class Reference</a>	3495
5.969.1 Detailed Description	3495
5.969.2 Member Function Documentation	3495
5.970 <a href="#">std::output_iterator_tag Struct Reference</a>	3496
5.970.1 Detailed Description	3496
5.971 <a href="#">std::overflow_error Class Reference</a>	3496
5.971.1 Detailed Description	3496
5.971.2 Member Function Documentation	3497

5.972 <a href="#">std::owner_less&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3497
5.972.1 Detailed Description . . . . .	3497
5.973 <a href="#">std::owner_less&lt; shared_ptr&lt; _Tp &gt; &gt; Struct Template Reference</a> . . . . .	3497
5.973.1 Detailed Description . . . . .	3498
5.973.2 Member Typedef Documentation . . . . .	3498
5.974 <a href="#">std::owner_less&lt; void &gt; Struct Template Reference</a> . . . . .	3498
5.974.1 Detailed Description . . . . .	3499
5.974.2 Member Typedef Documentation . . . . .	3499
5.975 <a href="#">std::owner_less&lt; weak_ptr&lt; _Tp &gt; &gt; Struct Template Reference</a> . . . . .	3500
5.975.1 Detailed Description . . . . .	3500
5.975.2 Member Typedef Documentation . . . . .	3500
5.976 <a href="#">std::packaged_task&lt; _Res(_ArgTypes...) &gt; Class Template Reference</a> . . . . .	3501
5.976.1 Detailed Description . . . . .	3502
5.977 <a href="#">std::pair&lt; _T1, _T2 &gt; Struct Template Reference</a> . . . . .	3502
5.977.1 Detailed Description . . . . .	3503
5.977.2 Member Typedef Documentation . . . . .	3504
5.977.3 Constructor & Destructor Documentation . . . . .	3504
5.977.4 Member Data Documentation . . . . .	3505
5.978 <a href="#">std::piecewise_constant_distribution&lt; _RealType &gt; Class Template Reference</a> . . . . .	3506
5.978.1 Detailed Description . . . . .	3507
5.978.2 Member Typedef Documentation . . . . .	3507
5.978.3 Member Function Documentation . . . . .	3507
5.978.4 Friends And Related Function Documentation . . . . .	3509
5.979 <a href="#">std::piecewise_constant_distribution&lt; _RealType &gt;::param_type Struct Reference</a> . . . . .	3511
5.979.1 Detailed Description . . . . .	3511
5.980 <a href="#">std::piecewise_construct_t Struct Reference</a> . . . . .	3511
5.980.1 Detailed Description . . . . .	3511
5.981 <a href="#">std::piecewise_linear_distribution&lt; _RealType &gt; Class Template Reference</a> . . . . .	3512
5.981.1 Detailed Description . . . . .	3513
5.981.2 Member Typedef Documentation . . . . .	3513
5.981.3 Member Function Documentation . . . . .	3513
5.981.4 Friends And Related Function Documentation . . . . .	3515
5.982 <a href="#">std::piecewise_linear_distribution&lt; _RealType &gt;::param_type Struct Reference</a> . . . . .	3517
5.982.1 Detailed Description . . . . .	3517
5.983 <a href="#">std::plus&lt; _Tp &gt; Struct Template Reference</a> . . . . .	3518
5.983.1 Detailed Description . . . . .	3518
5.983.2 Member Typedef Documentation . . . . .	3518
5.984 <a href="#">std::pointer_to_binary_function&lt; _Arg1, _Arg2, _Result &gt; Class Template Reference</a> . . . . .	3519
5.984.1 Detailed Description . . . . .	3520

5.984.2 Member Typedef Documentation . . . . .	3520
5.985 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference . . . . .	3521
5.985.1 Detailed Description . . . . .	3522
5.985.2 Member Typedef Documentation . . . . .	3522
5.986 std::pointer_traits< _Ptr > Struct Template Reference . . . . .	3522
5.986.1 Detailed Description . . . . .	3523
5.986.2 Member Typedef Documentation . . . . .	3523
5.987 std::pointer_traits< _Tp * > Struct Template Reference . . . . .	3524
5.987.1 Detailed Description . . . . .	3524
5.987.2 Member Typedef Documentation . . . . .	3524
5.987.3 Member Function Documentation . . . . .	3525
5.988 std::poisson_distribution< _IntType > Class Template Reference . . . . .	3526
5.988.1 Detailed Description . . . . .	3527
5.988.2 Member Typedef Documentation . . . . .	3527
5.988.3 Member Function Documentation . . . . .	3527
5.988.4 Friends And Related Function Documentation . . . . .	3529
5.989 std::poisson_distribution< _IntType >::param_type Struct Reference . . . . .	3531
5.989.1 Detailed Description . . . . .	3531
5.990 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference . . . . .	3531
5.990.1 Detailed Description . . . . .	3532
5.990.2 Constructor & Destructor Documentation . . . . .	3533
5.990.3 Member Function Documentation . . . . .	3534
5.991 std::promise< _Res > Class Template Reference . . . . .	3536
5.991.1 Detailed Description . . . . .	3536
5.992 std::promise< _Res & > Class Template Reference . . . . .	3537
5.992.1 Detailed Description . . . . .	3537
5.993 std::promise< void > Class Template Reference . . . . .	3537
5.993.1 Detailed Description . . . . .	3538
5.994 std::queue< _Tp, _Sequence > Class Template Reference . . . . .	3538
5.994.1 Detailed Description . . . . .	3539
5.994.2 Constructor & Destructor Documentation . . . . .	3539
5.994.3 Member Function Documentation . . . . .	3540
5.994.4 Member Data Documentation . . . . .	3542
5.995 std::random_access_iterator_tag Struct Reference . . . . .	3543
5.995.1 Detailed Description . . . . .	3543
5.996 std::random_device Class Reference . . . . .	3543
5.996.1 Detailed Description . . . . .	3544
5.996.2 Member Typedef Documentation . . . . .	3544
5.997 std::range_error Class Reference . . . . .	3545

5.997.1 Detailed Description . . . . .	3545
5.997.2 Member Function Documentation . . . . .	3545
5.998 std::rank< typename > Struct Template Reference . . . . .	3546
5.998.1 Detailed Description . . . . .	3546
5.999 std::ratio< _Num, _Den > Struct Template Reference . . . . .	3547
5.999.1 Detailed Description . . . . .	3547
5.1000 std::ratio_equal< _R1, _R2 > Struct Template Reference . . . . .	3547
5.1000.1 Detailed Description . . . . .	3548
5.1001 std::ratio_not_equal< _R1, _R2 > Struct Template Reference . . . . .	3548
5.1001.1 Detailed Description . . . . .	3549
5.1002 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference . . . . .	3549
5.1002.1 Detailed Description . . . . .	3550
5.1002.2 Member Typedef Documentation . . . . .	3550
5.1003 std::recursive_mutex Class Reference . . . . .	3551
5.1003.1 Detailed Description . . . . .	3552
5.1004 std::recursive_timed_mutex Class Reference . . . . .	3552
5.1004.1 Detailed Description . . . . .	3553
5.1005 std::reference_wrapper< _Tp > Class Template Reference . . . . .	3553
5.1005.1 Detailed Description . . . . .	3553
5.1006 std::regex_error Class Reference . . . . .	3554
5.1006.1 Detailed Description . . . . .	3554
5.1006.2 Constructor & Destructor Documentation . . . . .	3554
5.1006.3 Member Function Documentation . . . . .	3555
5.1007 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . . .	3555
5.1007.1 Detailed Description . . . . .	3556
5.1007.2 Constructor & Destructor Documentation . . . . .	3556
5.1007.3 Member Function Documentation . . . . .	3557
5.1008 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . . .	3559
5.1008.1 Detailed Description . . . . .	3560
5.1008.2 Constructor & Destructor Documentation . . . . .	3560
5.1008.3 Member Function Documentation . . . . .	3562
5.1009 std::regex_traits< _Ch_type > Class Template Reference . . . . .	3564
5.1009.1 Detailed Description . . . . .	3565
5.1009.2 Constructor & Destructor Documentation . . . . .	3565
5.1009.3 Member Function Documentation . . . . .	3566
5.1010 std::remove_all_extents< _Tp > Struct Template Reference . . . . .	3572
5.1010.1 Detailed Description . . . . .	3572
5.1011 std::remove_const< _Tp > Struct Template Reference . . . . .	3572
5.1011.1 Detailed Description . . . . .	3572

5.1012 <a href="#">std::remove_cv&lt;_Tp&gt; Struct Template Reference</a>	3573
5.1012.1 Detailed Description	3573
5.1013 <a href="#">std::remove_extent&lt;_Tp&gt; Struct Template Reference</a>	3573
5.1013.1 Detailed Description	3573
5.1014 <a href="#">std::remove_pointer&lt;_Tp&gt; Struct Template Reference</a>	3573
5.1014.1 Detailed Description	3574
5.1015 <a href="#">std::remove_reference&lt;_Tp&gt; Struct Template Reference</a>	3574
5.1015.1 Detailed Description	3574
5.1016 <a href="#">std::remove_volatile&lt;_Tp&gt; Struct Template Reference</a>	3574
5.1016.1 Detailed Description	3575
5.1017 <a href="#">std::result_of&lt;_Signature&gt; Class Template Reference</a>	3575
5.1017.1 Detailed Description	3575
5.1018 <a href="#">std::reverse_iterator&lt;_Iterator&gt; Class Template Reference</a>	3575
5.1018.1 Detailed Description	3576
5.1018.2 Member Typedef Documentation	3577
5.1018.3 Constructor & Destructor Documentation	3577
5.1018.4 Member Function Documentation	3578
5.1019 <a href="#">std::runtime_error Class Reference</a>	3582
5.1019.1 Detailed Description	3583
5.1019.2 Constructor & Destructor Documentation	3583
5.1019.3 Member Function Documentation	3583
5.1020 <a href="#">std::scoped_allocator_adaptor&lt;_OuterAlloc, _InnerAllocs&gt; Class Template Reference</a>	3583
5.1020.1 Detailed Description	3585
5.1021 <a href="#">std::seed_seq Class Reference</a>	3585
5.1021.1 Detailed Description	3586
5.1021.2 Member Typedef Documentation	3586
5.1021.3 Constructor & Destructor Documentation	3586
5.1022 <a href="#">std::set&lt;_Key, _Compare, _Alloc&gt; Class Template Reference</a>	3587
5.1022.1 Detailed Description	3589
5.1022.2 Member Typedef Documentation	3590
5.1022.3 Constructor & Destructor Documentation	3593
5.1022.4 Member Function Documentation	3598
5.1023 <a href="#">std::shared_future&lt;_Res&gt; Class Template Reference</a>	3617
5.1023.1 Detailed Description	3618
5.1023.2 Member Typedef Documentation	3618
5.1023.3 Constructor & Destructor Documentation	3619
5.1023.4 Member Function Documentation	3619
5.1024 <a href="#">std::shared_future&lt;_Res &amp;&gt; Class Template Reference</a>	3620
5.1024.1 Detailed Description	3621

5.1024.2 Member Typedef Documentation . . . . .	3622
5.1024.3 Constructor & Destructor Documentation . . . . .	3622
5.1024.4 Member Function Documentation . . . . .	3623
5.1025 std::shared_future< void > Class Template Reference . . . . .	3623
5.1025.1 Detailed Description . . . . .	3624
5.1025.2 Member Typedef Documentation . . . . .	3625
5.1025.3 Constructor & Destructor Documentation . . . . .	3625
5.1025.4 Member Function Documentation . . . . .	3626
5.1026 std::shared_lock< _Mutex > Class Template Reference . . . . .	3626
5.1026.1 Detailed Description . . . . .	3627
5.1027 std::shared_ptr< _Tp > Class Template Reference . . . . .	3627
5.1027.1 Detailed Description . . . . .	3628
5.1027.2 Constructor & Destructor Documentation . . . . .	3628
5.1027.3 Friends And Related Function Documentation . . . . .	3635
5.1028 std::shared_timed_mutex Class Reference . . . . .	3636
5.1028.1 Detailed Description . . . . .	3637
5.1029 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference . . . . .	3637
5.1029.1 Detailed Description . . . . .	3638
5.1029.2 Member Typedef Documentation . . . . .	3638
5.1029.3 Constructor & Destructor Documentation . . . . .	3638
5.1029.4 Member Function Documentation . . . . .	3640
5.1029.5 Friends And Related Function Documentation . . . . .	3642
5.1030 std::slice Class Reference . . . . .	3644
5.1030.1 Detailed Description . . . . .	3644
5.1031 std::slice_array< _Tp > Class Template Reference . . . . .	3644
5.1031.1 Detailed Description . . . . .	3645
5.1032 std::stack< _Tp, _Sequence > Class Template Reference . . . . .	3646
5.1032.1 Detailed Description . . . . .	3647
5.1032.2 Constructor & Destructor Documentation . . . . .	3647
5.1032.3 Member Function Documentation . . . . .	3648
5.1033 std::student_t_distribution< _RealType > Class Template Reference . . . . .	3649
5.1033.1 Detailed Description . . . . .	3650
5.1033.2 Member Typedef Documentation . . . . .	3651
5.1033.3 Member Function Documentation . . . . .	3651
5.1033.4 Friends And Related Function Documentation . . . . .	3653
5.1034 std::student_t_distribution< _RealType >::param_type Struct Reference . . . . .	3654
5.1034.1 Detailed Description . . . . .	3654
5.1035 std::sub_match< _Biter > Class Template Reference . . . . .	3655
5.1035.1 Detailed Description . . . . .	3656

5.1035.2 Member Typedef Documentation . . . . .	3656
5.1035.3 Member Function Documentation . . . . .	3656
5.1035.4 Member Data Documentation . . . . .	3659
5.1036 std::subtract_with_carry_engine< _UIntType, __w, __s, __r > Class Template Reference . . . . .	3659
5.1036.1 Detailed Description . . . . .	3660
5.1036.2 Member Typedef Documentation . . . . .	3661
5.1036.3 Constructor & Destructor Documentation . . . . .	3661
5.1036.4 Member Function Documentation . . . . .	3662
5.1036.5 Friends And Related Function Documentation . . . . .	3663
5.1037 std::system_error Class Reference . . . . .	3665
5.1037.1 Detailed Description . . . . .	3665
5.1037.2 Member Function Documentation . . . . .	3666
5.1038 std::thread Class Reference . . . . .	3666
5.1038.1 Detailed Description . . . . .	3667
5.1038.2 Member Function Documentation . . . . .	3667
5.1039 std::thread::id Class Reference . . . . .	3667
5.1039.1 Detailed Description . . . . .	3667
5.1040 std::time_base Class Reference . . . . .	3668
5.1040.1 Detailed Description . . . . .	3668
5.1041 std::time_get< _CharT, _InIter > Class Template Reference . . . . .	3669
5.1041.1 Detailed Description . . . . .	3670
5.1041.2 Member Typedef Documentation . . . . .	3671
5.1041.3 Constructor & Destructor Documentation . . . . .	3671
5.1041.4 Member Function Documentation . . . . .	3672
5.1041.5 Member Data Documentation . . . . .	3682
5.1042 std::time_get_byname< _CharT, _InIter > Class Template Reference . . . . .	3683
5.1042.1 Detailed Description . . . . .	3685
5.1042.2 Member Function Documentation . . . . .	3685
5.1042.3 Member Data Documentation . . . . .	3695
5.1043 std::time_put< _CharT, _OutIter > Class Template Reference . . . . .	3696
5.1043.1 Detailed Description . . . . .	3697
5.1043.2 Member Typedef Documentation . . . . .	3697
5.1043.3 Constructor & Destructor Documentation . . . . .	3698
5.1043.4 Member Function Documentation . . . . .	3699
5.1043.5 Member Data Documentation . . . . .	3701
5.1044 std::time_put_byname< _CharT, _OutIter > Class Template Reference . . . . .	3702
5.1044.1 Detailed Description . . . . .	3703
5.1044.2 Member Function Documentation . . . . .	3703
5.1044.3 Member Data Documentation . . . . .	3705



5.1045 std::timed_mutex Class Reference . . . . .	3706
5.1045.1 Detailed Description . . . . .	3706
5.1046 std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference . . . . .	3706
5.1046.1 Detailed Description . . . . .	3708
5.1046.2 Member Data Documentation . . . . .	3708
5.1047 std::tr2::__reflection_typelist< _Elements > Struct Template Reference . . . . .	3708
5.1047.1 Detailed Description . . . . .	3708
5.1048 std::tr2::__reflection_typelist< _First, _Rest... > Struct Template Reference . . . . .	3709
5.1048.1 Detailed Description . . . . .	3709
5.1049 std::tr2::__reflection_typelist<> Struct Template Reference . . . . .	3709
5.1049.1 Detailed Description . . . . .	3709
5.1050 std::tr2::bases< _Tp > Struct Template Reference . . . . .	3709
5.1050.1 Detailed Description . . . . .	3710
5.1051 std::tr2::bool_set Class Reference . . . . .	3710
5.1051.1 Detailed Description . . . . .	3711
5.1051.2 Constructor & Destructor Documentation . . . . .	3711
5.1051.3 Member Function Documentation . . . . .	3711
5.1052 std::tr2::direct_bases< _Tp > Struct Template Reference . . . . .	3712
5.1052.1 Detailed Description . . . . .	3713
5.1053 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference . . . . .	3713
5.1053.1 Detailed Description . . . . .	3717
5.1053.2 Constructor & Destructor Documentation . . . . .	3717
5.1053.3 Member Function Documentation . . . . .	3720
5.1054 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference . . . . .	3733
5.1054.1 Detailed Description . . . . .	3733
5.1055 std::try_to_lock_t Struct Reference . . . . .	3734
5.1055.1 Detailed Description . . . . .	3734
5.1056 std::tuple< _Elements > Class Template Reference . . . . .	3734
5.1056.1 Detailed Description . . . . .	3736
5.1057 std::tuple< _T1, _T2 > Class Template Reference . . . . .	3737
5.1057.1 Detailed Description . . . . .	3739
5.1058 std::tuple_element< _Int, _Tp > Struct Template Reference . . . . .	3739
5.1058.1 Detailed Description . . . . .	3739
5.1059 std::tuple_element< 0, std::pair< _Tp1, _Tp2 > > Struct Template Reference . . . . .	3740
5.1059.1 Detailed Description . . . . .	3740
5.1060 std::tuple_element< 0, tuple< _Head, _Tail... > > Struct Template Reference . . . . .	3740
5.1060.1 Detailed Description . . . . .	3740
5.1061 std::tuple_element< 1, std::pair< _Tp1, _Tp2 > > Struct Template Reference . . . . .	3740
5.1061.1 Detailed Description . . . . .	3741

5.1062 <code>std::tuple_element&lt; __i, tuple&lt; _Head, _Tail... &gt; &gt;</code> Struct Template Reference . . . . .	3741
5.1062.1 Detailed Description . . . . .	3741
5.1063 <code>std::tuple_element&lt; __i, tuple&lt;&gt; &gt;</code> Struct Template Reference . . . . .	3742
5.1063.1 Detailed Description . . . . .	3742
5.1064 <code>std::tuple_element&lt; _Int, ::array&lt; _Tp, _Nm &gt; &gt;</code> Struct Template Reference . . . . .	3742
5.1064.1 Detailed Description . . . . .	3742
5.1065 <code>std::tuple_element&lt; _Int, std::__debug::array&lt; _Tp, _Nm &gt; &gt;</code> Struct Template Reference . . . . .	3742
5.1065.1 Detailed Description . . . . .	3743
5.1066 <code>std::tuple_size&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3743
5.1066.1 Detailed Description . . . . .	3743
5.1067 <code>std::tuple_size&lt; std::__debug::array&lt; _Tp, _Nm &gt; &gt;</code> Struct Template Reference . . . . .	3743
5.1067.1 Detailed Description . . . . .	3744
5.1068 <code>std::tuple_size&lt; std::pair&lt; _Tp1, _Tp2 &gt; &gt;</code> Struct Template Reference . . . . .	3744
5.1068.1 Detailed Description . . . . .	3745
5.1069 <code>std::tuple_size&lt; tuple&lt; _Elements... &gt; &gt;</code> Struct Template Reference . . . . .	3745
5.1069.1 Detailed Description . . . . .	3746
5.1070 <code>std::tuple_size&lt;::array&lt; _Tp, _Nm &gt; &gt;</code> Struct Template Reference . . . . .	3746
5.1070.1 Detailed Description . . . . .	3747
5.1071 <code>std::type_index</code> Struct Reference . . . . .	3747
5.1071.1 Detailed Description . . . . .	3748
5.1072 <code>std::type_info</code> Class Reference . . . . .	3748
5.1072.1 Detailed Description . . . . .	3748
5.1072.2 Constructor & Destructor Documentation . . . . .	3749
5.1072.3 Member Function Documentation . . . . .	3749
5.1073 <code>std::unary_function&lt; _Arg, _Result &gt;</code> Struct Template Reference . . . . .	3749
5.1073.1 Detailed Description . . . . .	3750
5.1073.2 Member Typedef Documentation . . . . .	3750
5.1074 <code>std::unary_negate&lt; _Predicate &gt;</code> Class Template Reference . . . . .	3751
5.1074.1 Detailed Description . . . . .	3751
5.1074.2 Member Typedef Documentation . . . . .	3751
5.1075 <code>std::underflow_error</code> Class Reference . . . . .	3752
5.1075.1 Detailed Description . . . . .	3753
5.1075.2 Member Function Documentation . . . . .	3753
5.1076 <code>std::underlying_type&lt; _Tp &gt;</code> Struct Template Reference . . . . .	3753
5.1076.1 Detailed Description . . . . .	3753
5.1077 <code>std::uniform_int_distribution&lt; _IntType &gt;</code> Class Template Reference . . . . .	3753
5.1077.1 Detailed Description . . . . .	3754
5.1077.2 Member Typedef Documentation . . . . .	3754
5.1077.3 Constructor & Destructor Documentation . . . . .	3755

5.1077.4 Member Function Documentation . . . . .	3755
5.1077.5 Friends And Related Function Documentation . . . . .	3757
5.1078 std::uniform_int_distribution< _IntType >::param_type Struct Reference . . . . .	3757
5.1078.1 Detailed Description . . . . .	3757
5.1079 std::uniform_real_distribution< _RealType > Class Template Reference . . . . .	3758
5.1079.1 Detailed Description . . . . .	3758
5.1079.2 Member Typedef Documentation . . . . .	3759
5.1079.3 Constructor & Destructor Documentation . . . . .	3759
5.1079.4 Member Function Documentation . . . . .	3759
5.1079.5 Friends And Related Function Documentation . . . . .	3761
5.1080 std::uniform_real_distribution< _RealType >::param_type Struct Reference . . . . .	3761
5.1080.1 Detailed Description . . . . .	3762
5.1081 std::unique_lock< _Mutex > Class Template Reference . . . . .	3762
5.1081.1 Detailed Description . . . . .	3763
5.1082 std::unique_ptr< _Tp, _Dp > Class Template Reference . . . . .	3763
5.1082.1 Detailed Description . . . . .	3764
5.1082.2 Constructor & Destructor Documentation . . . . .	3764
5.1082.3 Member Function Documentation . . . . .	3767
5.1083 std::unique_ptr< _Tp[], _Dp > Class Template Reference . . . . .	3770
5.1083.1 Detailed Description . . . . .	3771
5.1083.2 Constructor & Destructor Documentation . . . . .	3771
5.1083.3 Member Function Documentation . . . . .	3774
5.1084 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3777
5.1084.1 Detailed Description . . . . .	3780
5.1084.2 Member Typedef Documentation . . . . .	3781
5.1084.3 Constructor & Destructor Documentation . . . . .	3785
5.1084.4 Member Function Documentation . . . . .	3788
5.1085 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3811
5.1085.1 Detailed Description . . . . .	3813
5.1085.2 Member Typedef Documentation . . . . .	3814
5.1085.3 Constructor & Destructor Documentation . . . . .	3818
5.1085.4 Member Function Documentation . . . . .	3821
5.1086 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3840
5.1086.1 Detailed Description . . . . .	3843
5.1086.2 Member Typedef Documentation . . . . .	3844
5.1086.3 Constructor & Destructor Documentation . . . . .	3847
5.1086.4 Member Function Documentation . . . . .	3850
5.1087 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . . .	3870
5.1087.1 Detailed Description . . . . .	3873

5.1087.2 Member Typedef Documentation . . . . .	3874
5.1087.3 Constructor & Destructor Documentation . . . . .	3877
5.1087.4 Member Function Documentation . . . . .	3880
5.1088 std::uses_allocator< _Tp, _Alloc > Struct Template Reference . . . . .	3899
5.1088.1 Detailed Description . . . . .	3899
5.1089 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference . . . . .	3900
5.1089.1 Detailed Description . . . . .	3900
5.1090 std::valarray< _Tp > Class Template Reference . . . . .	3901
5.1090.1 Detailed Description . . . . .	3903
5.1090.2 Constructor & Destructor Documentation . . . . .	3903
5.1091 std::vector< _Tp, _Alloc > Class Template Reference . . . . .	3903
5.1091.1 Detailed Description . . . . .	3907
5.1091.2 Constructor & Destructor Documentation . . . . .	3907
5.1091.3 Member Function Documentation . . . . .	3911
5.1092 std::vector< bool, _Alloc > Class Template Reference . . . . .	3930
5.1092.1 Detailed Description . . . . .	3933
5.1093 std::wbuffer_convert< _Codecvt, _Elem, _Tr > Class Template Reference . . . . .	3934
5.1093.1 Detailed Description . . . . .	3937
5.1093.2 Member Typedef Documentation . . . . .	3937
5.1093.3 Constructor & Destructor Documentation . . . . .	3938
5.1093.4 Member Function Documentation . . . . .	3939
5.1093.5 Member Data Documentation . . . . .	3954
5.1094 std::weak_ptr< _Tp > Class Template Reference . . . . .	3956
5.1094.1 Detailed Description . . . . .	3957
5.1095 std::weibull_distribution< _RealType > Class Template Reference . . . . .	3957
5.1095.1 Detailed Description . . . . .	3958
5.1095.2 Member Typedef Documentation . . . . .	3958
5.1095.3 Member Function Documentation . . . . .	3958
5.1095.4 Friends And Related Function Documentation . . . . .	3960
5.1096 std::weibull_distribution< _RealType >::param_type Struct Reference . . . . .	3961
5.1096.1 Detailed Description . . . . .	3961
5.1097 std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference . . . . .	3961
5.1097.1 Detailed Description . . . . .	3962
5.1097.2 Constructor & Destructor Documentation . . . . .	3962
5.1097.3 Member Function Documentation . . . . .	3964
<b>6 File Documentation . . . . .</b>	<b>3967</b>
6.1 algo.h File Reference . . . . .	3967
6.1.1 Detailed Description . . . . .	3976

6.2 <a href="#">allobase.h File Reference</a> . . . . .	3976
6.2.1 Detailed Description . . . . .	3978
6.3 <a href="#">algorithm File Reference</a> . . . . .	3978
6.3.1 Detailed Description . . . . .	3978
6.4 <a href="#">algorithm File Reference</a> . . . . .	3978
6.4.1 Detailed Description . . . . .	3980
6.5 <a href="#">algorithm File Reference</a> . . . . .	3980
6.5.1 Detailed Description . . . . .	3980
6.6 <a href="#">algorithm File Reference</a> . . . . .	3980
6.6.1 Detailed Description . . . . .	3980
6.6.2 Function Documentation . . . . .	3981
6.7 <a href="#">algorithmfwd.h File Reference</a> . . . . .	3981
6.7.1 Detailed Description . . . . .	3986
6.8 <a href="#">algorithmfwd.h File Reference</a> . . . . .	3986
6.8.1 Detailed Description . . . . .	3994
6.9 <a href="#">aligned_buffer.h File Reference</a> . . . . .	3995
6.9.1 Detailed Description . . . . .	3995
6.10 <a href="#">alloc_traits.h File Reference</a> . . . . .	3995
6.10.1 Detailed Description . . . . .	3996
6.11 <a href="#">alloc_traits.h File Reference</a> . . . . .	3996
6.11.1 Detailed Description . . . . .	3996
6.12 <a href="#">allocated_ptr.h File Reference</a> . . . . .	3996
6.12.1 Detailed Description . . . . .	3996
6.13 <a href="#">allocator.h File Reference</a> . . . . .	3997
6.13.1 Detailed Description . . . . .	3997
6.14 <a href="#">any File Reference</a> . . . . .	3997
6.14.1 Detailed Description . . . . .	3998
6.15 <a href="#">array File Reference</a> . . . . .	3998
6.15.1 Detailed Description . . . . .	3999
6.16 <a href="#">array File Reference</a> . . . . .	3999
6.16.1 Detailed Description . . . . .	4000
6.17 <a href="#">array File Reference</a> . . . . .	4000
6.17.1 Detailed Description . . . . .	4001
6.18 <a href="#">array_allocator.h File Reference</a> . . . . .	4001
6.18.1 Detailed Description . . . . .	4001
6.19 <a href="#">assertions.h File Reference</a> . . . . .	4001
6.19.1 Detailed Description . . . . .	4002
6.20 <a href="#">assoc_container.hpp File Reference</a> . . . . .	4002
6.20.1 Detailed Description . . . . .	4002

---

6.21 atomic File Reference	4003
6.21.1 Detailed Description	4007
6.22 atomic_base.h File Reference	4007
6.22.1 Detailed Description	4008
6.23 atomic_futex.h File Reference	4008
6.23.1 Detailed Description	4008
6.24 atomic_lockfree_defines.h File Reference	4008
6.24.1 Detailed Description	4008
6.25 atomic_word.h File Reference	4008
6.25.1 Detailed Description	4009
6.26 atomicity.h File Reference	4009
6.26.1 Detailed Description	4009
6.27 auto_ptr.h File Reference	4009
6.27.1 Detailed Description	4010
6.28 backward_warning.h File Reference	4010
6.28.1 Detailed Description	4010
6.29 balanced_quicksort.h File Reference	4010
6.29.1 Detailed Description	4011
6.30 base.h File Reference	4011
6.30.1 Detailed Description	4011
6.31 base.h File Reference	4011
6.31.1 Detailed Description	4012
6.32 basic_file.h File Reference	4012
6.32.1 Detailed Description	4012
6.33 basic_ios.h File Reference	4012
6.33.1 Detailed Description	4013
6.34 basic_ios.tcc File Reference	4013
6.34.1 Detailed Description	4013
6.35 basic_iterator.h File Reference	4013
6.35.1 Detailed Description	4013
6.36 basic_string.h File Reference	4013
6.36.1 Detailed Description	4016
6.37 basic_string.tcc File Reference	4016
6.37.1 Detailed Description	4017
6.38 bin_search_tree_.hpp File Reference	4017
6.38.1 Detailed Description	4017
6.39 binary_heap_.hpp File Reference	4017
6.39.1 Detailed Description	4018
6.40 binders.h File Reference	4018

6.40.1 Detailed Description . . . . .	4018
6.41 binomial_heap_.hpp File Reference . . . . .	4018
6.41.1 Detailed Description . . . . .	4019
6.42 binomial_heap_base_.hpp File Reference . . . . .	4019
6.42.1 Detailed Description . . . . .	4019
6.43 bitmap_allocator.h File Reference . . . . .	4019
6.43.1 Detailed Description . . . . .	4020
6.43.2 Macro Definition Documentation . . . . .	4020
6.44 bitset File Reference . . . . .	4021
6.44.1 Detailed Description . . . . .	4021
6.45 bitset File Reference . . . . .	4022
6.45.1 Detailed Description . . . . .	4022
6.46 bitset File Reference . . . . .	4022
6.46.1 Detailed Description . . . . .	4023
6.47 bool_set File Reference . . . . .	4023
6.47.1 Detailed Description . . . . .	4024
6.48 bool_set.tcc File Reference . . . . .	4024
6.48.1 Detailed Description . . . . .	4024
6.49 boost_concept_check.h File Reference . . . . .	4024
6.49.1 Detailed Description . . . . .	4025
6.50 branch_policy.hpp File Reference . . . . .	4025
6.50.1 Detailed Description . . . . .	4025
6.51 c++0x_warning.h File Reference . . . . .	4025
6.51.1 Detailed Description . . . . .	4025
6.52 c++allocator.h File Reference . . . . .	4026
6.52.1 Detailed Description . . . . .	4026
6.53 c++config.h File Reference . . . . .	4026
6.53.1 Detailed Description . . . . .	4032
6.54 c++io.h File Reference . . . . .	4032
6.54.1 Detailed Description . . . . .	4032
6.55 c++locale.h File Reference . . . . .	4032
6.55.1 Detailed Description . . . . .	4033
6.56 c++locale_internal.h File Reference . . . . .	4033
6.56.1 Detailed Description . . . . .	4033
6.57 cassert File Reference . . . . .	4033
6.57.1 Detailed Description . . . . .	4033
6.58 cast.h File Reference . . . . .	4033
6.58.1 Detailed Description . . . . .	4034
6.59 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference . . . . .	4034

---

6.59.1 Detailed Description . . . . .	4034
6.60 cc_ht_map_.hpp File Reference . . . . .	4034
6.60.1 Detailed Description . . . . .	4035
6.61 ccomplex File Reference . . . . .	4035
6.61.1 Detailed Description . . . . .	4035
6.62 ccomplex File Reference . . . . .	4035
6.62.1 Detailed Description . . . . .	4035
6.63 cctype File Reference . . . . .	4035
6.63.1 Detailed Description . . . . .	4036
6.64 cctype File Reference . . . . .	4036
6.64.1 Detailed Description . . . . .	4036
6.65 cerrno File Reference . . . . .	4036
6.65.1 Detailed Description . . . . .	4036
6.66 cfenv File Reference . . . . .	4036
6.66.1 Detailed Description . . . . .	4037
6.67 cfenv File Reference . . . . .	4037
6.67.1 Detailed Description . . . . .	4037
6.68 cfloat File Reference . . . . .	4037
6.68.1 Detailed Description . . . . .	4037
6.69 cfloat File Reference . . . . .	4037
6.69.1 Detailed Description . . . . .	4037
6.70 char_traits.h File Reference . . . . .	4038
6.70.1 Detailed Description . . . . .	4038
6.71 checkers.h File Reference . . . . .	4038
6.71.1 Detailed Description . . . . .	4038
6.72 chrono File Reference . . . . .	4039
6.72.1 Detailed Description . . . . .	4042
6.72.2 Typedef Documentation . . . . .	4042
6.73 chrono File Reference . . . . .	4042
6.73.1 Detailed Description . . . . .	4042
6.74 cinttypes File Reference . . . . .	4042
6.74.1 Detailed Description . . . . .	4043
6.75 cinttypes File Reference . . . . .	4043
6.75.1 Detailed Description . . . . .	4043
6.76 ciso646 File Reference . . . . .	4043
6.76.1 Detailed Description . . . . .	4043
6.77 climits File Reference . . . . .	4043
6.77.1 Detailed Description . . . . .	4043
6.78 climits File Reference . . . . .	4044



6.78.1 Detailed Description . . . . .	4044
6.79 locale File Reference . . . . .	4044
6.79.1 Detailed Description . . . . .	4044
6.80 cmath File Reference . . . . .	4044
6.80.1 Detailed Description . . . . .	4046
6.81 cmath File Reference . . . . .	4047
6.81.1 Detailed Description . . . . .	4047
6.82 cmath File Reference . . . . .	4047
6.82.1 Detailed Description . . . . .	4049
6.83 cmp_fn_imps.hpp File Reference . . . . .	4049
6.83.1 Detailed Description . . . . .	4049
6.84 codecvt File Reference . . . . .	4049
6.84.1 Detailed Description . . . . .	4050
6.85 codecvt.h File Reference . . . . .	4050
6.85.1 Detailed Description . . . . .	4050
6.86 codecvt_specializations.h File Reference . . . . .	4050
6.86.1 Detailed Description . . . . .	4051
6.87 compatibility.h File Reference . . . . .	4051
6.87.1 Detailed Description . . . . .	4051
6.88 compatibility.h File Reference . . . . .	4051
6.88.1 Detailed Description . . . . .	4051
6.89 compiletime_settings.h File Reference . . . . .	4052
6.89.1 Detailed Description . . . . .	4052
6.89.2 Macro Definition Documentation . . . . .	4052
6.90 complex File Reference . . . . .	4053
6.90.1 Detailed Description . . . . .	4058
6.91 complex File Reference . . . . .	4058
6.91.1 Detailed Description . . . . .	4058
6.92 complex.h File Reference . . . . .	4059
6.92.1 Detailed Description . . . . .	4059
6.93 concept_check.h File Reference . . . . .	4059
6.93.1 Detailed Description . . . . .	4059
6.94 concurrence.h File Reference . . . . .	4059
6.94.1 Detailed Description . . . . .	4060
6.95 cond_dealtor.hpp File Reference . . . . .	4060
6.95.1 Detailed Description . . . . .	4060
6.96 cond_key_dtor_entry_dealtor.hpp File Reference . . . . .	4060
6.96.1 Detailed Description . . . . .	4060
6.97 condition_variable File Reference . . . . .	4060

---

6.97.1 Detailed Description . . . . .	4061
6.98 <code>const_iterator.hpp</code> File Reference . . . . .	4061
6.98.1 Detailed Description . . . . .	4061
6.99 <code>const_iterator.hpp</code> File Reference . . . . .	4062
6.99.1 Detailed Description . . . . .	4062
6.100 <code>const_iterator.hpp</code> File Reference . . . . .	4062
6.100.1 Detailed Description . . . . .	4062
6.101 <code>constructor_destructor_fn_imps.hpp</code> File Reference . . . . .	4062
6.101.1 Detailed Description . . . . .	4062
6.102 <code>constructor_destructor_fn_imps.hpp</code> File Reference . . . . .	4062
6.102.1 Detailed Description . . . . .	4062
6.103 <code>constructor_destructor_fn_imps.hpp</code> File Reference . . . . .	4063
6.104 <code>constructor_destructor_no_store_hash_fn_imps.hpp</code> File Reference . . . . .	4063
6.104.1 Detailed Description . . . . .	4063
6.105 <code>constructor_destructor_no_store_hash_fn_imps.hpp</code> File Reference . . . . .	4063
6.105.1 Detailed Description . . . . .	4063
6.106 <code>constructor_destructor_store_hash_fn_imps.hpp</code> File Reference . . . . .	4063
6.106.1 Detailed Description . . . . .	4063
6.107 <code>constructor_destructor_store_hash_fn_imps.hpp</code> File Reference . . . . .	4063
6.107.1 Detailed Description . . . . .	4063
6.108 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4063
6.108.1 Detailed Description . . . . .	4063
6.109 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4063
6.109.1 Detailed Description . . . . .	4063
6.110 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4064
6.110.1 Detailed Description . . . . .	4064
6.111 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4064
6.111.1 Detailed Description . . . . .	4064
6.112 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4064
6.112.1 Detailed Description . . . . .	4064
6.113 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4064
6.113.1 Detailed Description . . . . .	4064
6.114 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4064
6.114.1 Detailed Description . . . . .	4064
6.115 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4064
6.115.1 Detailed Description . . . . .	4064
6.116 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4065
6.116.1 Detailed Description . . . . .	4065
6.117 <code>constructors_destructor_fn_imps.hpp</code> File Reference . . . . .	4065

6.117.1 Detailed Description . . . . .	4065
6.118 constructors_destructor_fn_imps.hpp File Reference . . . . .	4065
6.118.1 Detailed Description . . . . .	4065
6.119 constructors_destructor_fn_imps.hpp File Reference . . . . .	4065
6.119.1 Detailed Description . . . . .	4065
6.120 container_base_dispatch.hpp File Reference . . . . .	4065
6.120.1 Detailed Description . . . . .	4066
6.121 cpp_type_traits.h File Reference . . . . .	4066
6.121.1 Detailed Description . . . . .	4066
6.122 cpu_defines.h File Reference . . . . .	4067
6.122.1 Detailed Description . . . . .	4067
6.123 csetjmp File Reference . . . . .	4067
6.123.1 Detailed Description . . . . .	4067
6.124 csignal File Reference . . . . .	4067
6.124.1 Detailed Description . . . . .	4067
6.125 cstdalign File Reference . . . . .	4068
6.125.1 Detailed Description . . . . .	4068
6.126 cstdarg File Reference . . . . .	4068
6.126.1 Detailed Description . . . . .	4068
6.127 cstdarg File Reference . . . . .	4068
6.127.1 Detailed Description . . . . .	4068
6.128 cstdbool File Reference . . . . .	4069
6.128.1 Detailed Description . . . . .	4069
6.129 cstdbool File Reference . . . . .	4069
6.129.1 Detailed Description . . . . .	4069
6.130 cstddef File Reference . . . . .	4069
6.130.1 Detailed Description . . . . .	4069
6.131 cstdint File Reference . . . . .	4069
6.131.1 Detailed Description . . . . .	4070
6.132 cstdint File Reference . . . . .	4070
6.132.1 Detailed Description . . . . .	4070
6.133 cstdio File Reference . . . . .	4070
6.133.1 Detailed Description . . . . .	4070
6.134 cstdio File Reference . . . . .	4071
6.134.1 Detailed Description . . . . .	4071
6.135 cstdlib File Reference . . . . .	4071
6.135.1 Detailed Description . . . . .	4071
6.136 cstdlib File Reference . . . . .	4071
6.136.1 Detailed Description . . . . .	4072

6.137 cstring File Reference . . . . .	4072
6.137.1 Detailed Description . . . . .	4072
6.138 ctgmath File Reference . . . . .	4072
6.138.1 Detailed Description . . . . .	4072
6.139 ctgmath File Reference . . . . .	4073
6.139.1 Detailed Description . . . . .	4073
6.140 ctime File Reference . . . . .	4073
6.140.1 Detailed Description . . . . .	4073
6.141 ctime File Reference . . . . .	4073
6.141.1 Detailed Description . . . . .	4073
6.142 ctype_base.h File Reference . . . . .	4073
6.142.1 Detailed Description . . . . .	4074
6.143 ctype_inline.h File Reference . . . . .	4074
6.143.1 Detailed Description . . . . .	4074
6.144 cuchar File Reference . . . . .	4074
6.144.1 Detailed Description . . . . .	4074
6.145 cwchar File Reference . . . . .	4074
6.145.1 Detailed Description . . . . .	4075
6.146 cwchar File Reference . . . . .	4075
6.146.1 Detailed Description . . . . .	4075
6.147 cwctype File Reference . . . . .	4075
6.147.1 Detailed Description . . . . .	4076
6.148 cwctype File Reference . . . . .	4076
6.148.1 Detailed Description . . . . .	4076
6.149 cxxabi.h File Reference . . . . .	4076
6.149.1 Detailed Description . . . . .	4078
6.149.2 Function Documentation . . . . .	4078
6.150 cxxabi_forced.h File Reference . . . . .	4079
6.150.1 Detailed Description . . . . .	4079
6.151 cxxabi_init_exception.h File Reference . . . . .	4079
6.151.1 Detailed Description . . . . .	4079
6.152 cxxabi_tweaks.h File Reference . . . . .	4079
6.152.1 Detailed Description . . . . .	4080
6.153 debug.h File Reference . . . . .	4080
6.153.1 Detailed Description . . . . .	4080
6.154 debug_allocator.h File Reference . . . . .	4081
6.154.1 Detailed Description . . . . .	4081
6.155 debug_fn_imps.hpp File Reference . . . . .	4081
6.155.1 Detailed Description . . . . .	4081

6.156 debug_fn_imps.hpp File Reference . . . . .	4081
6.156.1 Detailed Description . . . . .	4081
6.157 debug_fn_imps.hpp File Reference . . . . .	4081
6.157.1 Detailed Description . . . . .	4081
6.158 debug_fn_imps.hpp File Reference . . . . .	4082
6.158.1 Detailed Description . . . . .	4082
6.159 debug_fn_imps.hpp File Reference . . . . .	4082
6.159.1 Detailed Description . . . . .	4082
6.160 debug_fn_imps.hpp File Reference . . . . .	4082
6.160.1 Detailed Description . . . . .	4082
6.161 debug_fn_imps.hpp File Reference . . . . .	4082
6.161.1 Detailed Description . . . . .	4082
6.162 debug_fn_imps.hpp File Reference . . . . .	4082
6.162.1 Detailed Description . . . . .	4082
6.163 debug_fn_imps.hpp File Reference . . . . .	4082
6.163.1 Detailed Description . . . . .	4082
6.164 debug_fn_imps.hpp File Reference . . . . .	4083
6.164.1 Detailed Description . . . . .	4083
6.165 debug_fn_imps.hpp File Reference . . . . .	4083
6.165.1 Detailed Description . . . . .	4083
6.166 debug_fn_imps.hpp File Reference . . . . .	4083
6.166.1 Detailed Description . . . . .	4083
6.167 debug_fn_imps.hpp File Reference . . . . .	4083
6.167.1 Detailed Description . . . . .	4083
6.168 debug_fn_imps.hpp File Reference . . . . .	4083
6.168.1 Detailed Description . . . . .	4083
6.169 debug_fn_imps.hpp File Reference . . . . .	4083
6.169.1 Detailed Description . . . . .	4083
6.170 debug_map_base.hpp File Reference . . . . .	4084
6.170.1 Detailed Description . . . . .	4084
6.171 debug_no_store_hash_fn_imps.hpp File Reference . . . . .	4084
6.171.1 Detailed Description . . . . .	4084
6.172 debug_no_store_hash_fn_imps.hpp File Reference . . . . .	4084
6.172.1 Detailed Description . . . . .	4084
6.173 debug_store_hash_fn_imps.hpp File Reference . . . . .	4084
6.173.1 Detailed Description . . . . .	4084
6.174 debug_store_hash_fn_imps.hpp File Reference . . . . .	4084
6.174.1 Detailed Description . . . . .	4084
6.175 decimal File Reference . . . . .	4084

6.175.1 Detailed Description . . . . .	4094
6.176 deque File Reference . . . . .	4094
6.176.1 Detailed Description . . . . .	4094
6.177 deque File Reference . . . . .	4094
6.177.1 Detailed Description . . . . .	4095
6.178 deque File Reference . . . . .	4095
6.178.1 Detailed Description . . . . .	4095
6.179 deque File Reference . . . . .	4095
6.179.1 Detailed Description . . . . .	4096
6.180 deque.tcc File Reference . . . . .	4096
6.180.1 Detailed Description . . . . .	4097
6.181 direct_mask_range_hashing_imp.hpp File Reference . . . . .	4097
6.181.1 Detailed Description . . . . .	4097
6.182 direct_mod_range_hashing_imp.hpp File Reference . . . . .	4097
6.182.1 Detailed Description . . . . .	4097
6.183 dynamic_bitset File Reference . . . . .	4097
6.183.1 Detailed Description . . . . .	4098
6.184 dynamic_bitset.tcc File Reference . . . . .	4098
6.184.1 Detailed Description . . . . .	4099
6.185 enable_special_members.h File Reference . . . . .	4099
6.185.1 Detailed Description . . . . .	4099
6.186 enc_filebuf.h File Reference . . . . .	4099
6.186.1 Detailed Description . . . . .	4100
6.187 entry_cmp.hpp File Reference . . . . .	4100
6.187.1 Detailed Description . . . . .	4100
6.188 entry_list_fn_imps.hpp File Reference . . . . .	4100
6.188.1 Detailed Description . . . . .	4100
6.189 entry_metadata_base.hpp File Reference . . . . .	4100
6.189.1 Detailed Description . . . . .	4100
6.190 entry_pred.hpp File Reference . . . . .	4101
6.190.1 Detailed Description . . . . .	4101
6.191 eq_by_less.hpp File Reference . . . . .	4101
6.191.1 Detailed Description . . . . .	4101
6.192 equally_split.h File Reference . . . . .	4101
6.192.1 Detailed Description . . . . .	4102
6.193 erase_fn_imps.hpp File Reference . . . . .	4102
6.193.1 Detailed Description . . . . .	4102
6.194 erase_fn_imps.hpp File Reference . . . . .	4102
6.194.1 Detailed Description . . . . .	4102

6.195 erase_fn_imps.hpp File Reference . . . . .	4102
6.195.1 Detailed Description . . . . .	4102
6.196 erase_fn_imps.hpp File Reference . . . . .	4102
6.196.1 Detailed Description . . . . .	4102
6.197 erase_fn_imps.hpp File Reference . . . . .	4102
6.197.1 Detailed Description . . . . .	4102
6.198 erase_fn_imps.hpp File Reference . . . . .	4102
6.198.1 Detailed Description . . . . .	4102
6.199 erase_fn_imps.hpp File Reference . . . . .	4103
6.199.1 Detailed Description . . . . .	4103
6.200 erase_fn_imps.hpp File Reference . . . . .	4103
6.200.1 Detailed Description . . . . .	4103
6.201 erase_fn_imps.hpp File Reference . . . . .	4103
6.201.1 Detailed Description . . . . .	4103
6.202 erase_fn_imps.hpp File Reference . . . . .	4103
6.202.1 Detailed Description . . . . .	4103
6.203 erase_fn_imps.hpp File Reference . . . . .	4103
6.203.1 Detailed Description . . . . .	4103
6.204 erase_fn_imps.hpp File Reference . . . . .	4103
6.204.1 Detailed Description . . . . .	4103
6.205 erase_fn_imps.hpp File Reference . . . . .	4104
6.205.1 Detailed Description . . . . .	4104
6.206 erase_fn_imps.hpp File Reference . . . . .	4104
6.206.1 Detailed Description . . . . .	4104
6.207 erase_if.h File Reference . . . . .	4104
6.207.1 Detailed Description . . . . .	4104
6.208 erase_no_store_hash_fn_imps.hpp File Reference . . . . .	4104
6.208.1 Detailed Description . . . . .	4104
6.209 erase_no_store_hash_fn_imps.hpp File Reference . . . . .	4104
6.209.1 Detailed Description . . . . .	4104
6.210 erase_store_hash_fn_imps.hpp File Reference . . . . .	4105
6.210.1 Detailed Description . . . . .	4105
6.211 erase_store_hash_fn_imps.hpp File Reference . . . . .	4105
6.211.1 Detailed Description . . . . .	4105
6.212 error_constants.h File Reference . . . . .	4105
6.212.1 Detailed Description . . . . .	4105
6.213 exception File Reference . . . . .	4106
6.213.1 Detailed Description . . . . .	4106
6.214 exception.h File Reference . . . . .	4106

---

6.214.1 Detailed Description . . . . .	4107
6.215 exception.hpp File Reference . . . . .	4107
6.215.1 Detailed Description . . . . .	4107
6.216 exception_defines.h File Reference . . . . .	4107
6.216.1 Detailed Description . . . . .	4108
6.217 exception_ptr.h File Reference . . . . .	4108
6.217.1 Detailed Description . . . . .	4108
6.218 extc++.h File Reference . . . . .	4108
6.218.1 Detailed Description . . . . .	4108
6.219 extptr_allocator.h File Reference . . . . .	4108
6.219.1 Detailed Description . . . . .	4109
6.220 features.h File Reference . . . . .	4109
6.220.1 Detailed Description . . . . .	4109
6.220.2 Macro Definition Documentation . . . . .	4109
6.221 fenv.h File Reference . . . . .	4112
6.221.1 Detailed Description . . . . .	4112
6.222 filesystem File Reference . . . . .	4112
6.222.1 Detailed Description . . . . .	4112
6.223 find.h File Reference . . . . .	4112
6.223.1 Detailed Description . . . . .	4113
6.224 find_fn_imps.hpp File Reference . . . . .	4113
6.224.1 Detailed Description . . . . .	4113
6.225 find_fn_imps.hpp File Reference . . . . .	4113
6.225.1 Detailed Description . . . . .	4113
6.226 find_fn_imps.hpp File Reference . . . . .	4113
6.226.1 Detailed Description . . . . .	4113
6.227 find_fn_imps.hpp File Reference . . . . .	4113
6.227.1 Detailed Description . . . . .	4113
6.228 find_fn_imps.hpp File Reference . . . . .	4113
6.228.1 Detailed Description . . . . .	4113
6.229 find_fn_imps.hpp File Reference . . . . .	4113
6.229.1 Detailed Description . . . . .	4113
6.230 find_fn_imps.hpp File Reference . . . . .	4114
6.230.1 Detailed Description . . . . .	4114
6.231 find_fn_imps.hpp File Reference . . . . .	4114
6.231.1 Detailed Description . . . . .	4114
6.232 find_fn_imps.hpp File Reference . . . . .	4114
6.232.1 Detailed Description . . . . .	4114
6.233 find_fn_imps.hpp File Reference . . . . .	4114



6.233.1 Detailed Description . . . . .	4114
6.234 find_fn_imps.hpp File Reference . . . . .	4114
6.234.1 Detailed Description . . . . .	4114
6.235 find_no_store_hash_fn_imps.hpp File Reference . . . . .	4114
6.235.1 Detailed Description . . . . .	4114
6.236 find_selectors.h File Reference . . . . .	4115
6.236.1 Detailed Description . . . . .	4115
6.237 find_store_hash_fn_imps.hpp File Reference . . . . .	4115
6.237.1 Detailed Description . . . . .	4115
6.238 find_store_hash_fn_imps.hpp File Reference . . . . .	4115
6.238.1 Detailed Description . . . . .	4115
6.239 for_each.h File Reference . . . . .	4115
6.239.1 Detailed Description . . . . .	4116
6.240 for_each_selectors.h File Reference . . . . .	4116
6.240.1 Detailed Description . . . . .	4116
6.241 formatter.h File Reference . . . . .	4116
6.241.1 Detailed Description . . . . .	4117
6.242 forward_list File Reference . . . . .	4117
6.242.1 Detailed Description . . . . .	4117
6.243 forward_list File Reference . . . . .	4118
6.243.1 Detailed Description . . . . .	4118
6.244 forward_list File Reference . . . . .	4119
6.244.1 Detailed Description . . . . .	4119
6.245 forward_list File Reference . . . . .	4119
6.245.1 Detailed Description . . . . .	4120
6.246 forward_list.h File Reference . . . . .	4120
6.246.1 Detailed Description . . . . .	4121
6.247 forward_list.tcc File Reference . . . . .	4121
6.247.1 Detailed Description . . . . .	4121
6.248 fs_dir.h File Reference . . . . .	4122
6.248.1 Detailed Description . . . . .	4122
6.249 fs_dir.h File Reference . . . . .	4122
6.249.1 Detailed Description . . . . .	4122
6.250 fs_fwd.h File Reference . . . . .	4122
6.250.1 Detailed Description . . . . .	4122
6.251 fs_fwd.h File Reference . . . . .	4123
6.251.1 Detailed Description . . . . .	4124
6.252 fs_path.h File Reference . . . . .	4124
6.252.1 Detailed Description . . . . .	4124

6.253 fs_path.h File Reference . . . . .	4125
6.253.1 Detailed Description . . . . .	4125
6.253.2 Function Documentation . . . . .	4125
6.254 fstream File Reference . . . . .	4129
6.254.1 Detailed Description . . . . .	4129
6.255 fstream.tcc File Reference . . . . .	4129
6.255.1 Detailed Description . . . . .	4130
6.256 funtexcept.h File Reference . . . . .	4130
6.256.1 Detailed Description . . . . .	4130
6.257 functional File Reference . . . . .	4131
6.257.1 Detailed Description . . . . .	4132
6.258 functional File Reference . . . . .	4133
6.258.1 Detailed Description . . . . .	4134
6.259 functional File Reference . . . . .	4134
6.259.1 Detailed Description . . . . .	4135
6.259.2 Function Documentation . . . . .	4135
6.259.3 Variable Documentation . . . . .	4136
6.260 functional_hash.h File Reference . . . . .	4136
6.260.1 Detailed Description . . . . .	4137
6.261 functions.h File Reference . . . . .	4137
6.261.1 Detailed Description . . . . .	4139
6.262 future File Reference . . . . .	4139
6.262.1 Detailed Description . . . . .	4140
6.263 gp_ht_map_.hpp File Reference . . . . .	4141
6.263.1 Detailed Description . . . . .	4141
6.264 gslice.h File Reference . . . . .	4141
6.264.1 Detailed Description . . . . .	4141
6.265 gslice_array.h File Reference . . . . .	4142
6.265.1 Detailed Description . . . . .	4142
6.266 hash_bytes.h File Reference . . . . .	4142
6.266.1 Detailed Description . . . . .	4142
6.267 hash_eq_fn.hpp File Reference . . . . .	4142
6.267.1 Detailed Description . . . . .	4143
6.268 hash_exponential_size_policy_imp.hpp File Reference . . . . .	4143
6.268.1 Detailed Description . . . . .	4143
6.269 hash_fun.h File Reference . . . . .	4143
6.269.1 Detailed Description . . . . .	4143
6.270 hash_load_check_resize_trigger_imp.hpp File Reference . . . . .	4143
6.270.1 Detailed Description . . . . .	4143

---

6.271 hash_load_check_resize_trigger_size_base.hpp File Reference	4144
6.271.1 Detailed Description	4144
6.272 hash_map File Reference	4144
6.272.1 Detailed Description	4145
6.273 hash_policy.hpp File Reference	4145
6.273.1 Detailed Description	4146
6.274 hash_prime_size_policy_imp.hpp File Reference	4146
6.274.1 Detailed Description	4146
6.275 hash_set File Reference	4146
6.275.1 Detailed Description	4147
6.276 hash_standard_resize_policy_imp.hpp File Reference	4147
6.276.1 Detailed Description	4147
6.277 hashtable.h File Reference	4147
6.277.1 Detailed Description	4148
6.278 hashtable.h File Reference	4148
6.278.1 Detailed Description	4148
6.279 hashtable_policy.h File Reference	4148
6.279.1 Detailed Description	4150
6.280 helper_functions.h File Reference	4150
6.280.1 Detailed Description	4151
6.281 indirect_array.h File Reference	4151
6.281.1 Detailed Description	4152
6.282 info_fn_imps.hpp File Reference	4152
6.282.1 Detailed Description	4152
6.283 info_fn_imps.hpp File Reference	4152
6.283.1 Detailed Description	4152
6.284 info_fn_imps.hpp File Reference	4152
6.284.1 Detailed Description	4152
6.285 info_fn_imps.hpp File Reference	4152
6.285.1 Detailed Description	4152
6.286 info_fn_imps.hpp File Reference	4152
6.286.1 Detailed Description	4152
6.287 info_fn_imps.hpp File Reference	4152
6.287.1 Detailed Description	4152
6.288 info_fn_imps.hpp File Reference	4153
6.288.1 Detailed Description	4153
6.289 info_fn_imps.hpp File Reference	4153
6.289.1 Detailed Description	4153
6.290 info_fn_imps.hpp File Reference	4153

---

6.290.1 Detailed Description . . . . .	4153
6.291 info_fn_imps.hpp File Reference . . . . .	4153
6.291.1 Detailed Description . . . . .	4153
6.292 initializer_list File Reference . . . . .	4153
6.292.1 Detailed Description . . . . .	4154
6.293 insert_fn_imps.hpp File Reference . . . . .	4154
6.293.1 Detailed Description . . . . .	4154
6.294 insert_fn_imps.hpp File Reference . . . . .	4154
6.294.1 Detailed Description . . . . .	4154
6.295 insert_fn_imps.hpp File Reference . . . . .	4154
6.295.1 Detailed Description . . . . .	4154
6.296 insert_fn_imps.hpp File Reference . . . . .	4154
6.296.1 Detailed Description . . . . .	4154
6.297 insert_fn_imps.hpp File Reference . . . . .	4154
6.297.1 Detailed Description . . . . .	4154
6.298 insert_fn_imps.hpp File Reference . . . . .	4154
6.298.1 Detailed Description . . . . .	4154
6.299 insert_fn_imps.hpp File Reference . . . . .	4155
6.299.1 Detailed Description . . . . .	4155
6.300 insert_fn_imps.hpp File Reference . . . . .	4155
6.300.1 Detailed Description . . . . .	4155
6.301 insert_fn_imps.hpp File Reference . . . . .	4155
6.301.1 Detailed Description . . . . .	4155
6.302 insert_fn_imps.hpp File Reference . . . . .	4155
6.302.1 Detailed Description . . . . .	4155
6.303 insert_fn_imps.hpp File Reference . . . . .	4155
6.303.1 Detailed Description . . . . .	4155
6.304 insert_fn_imps.hpp File Reference . . . . .	4155
6.304.1 Detailed Description . . . . .	4155
6.305 insert_fn_imps.hpp File Reference . . . . .	4156
6.305.1 Detailed Description . . . . .	4156
6.306 insert_join_fn_imps.hpp File Reference . . . . .	4156
6.306.1 Detailed Description . . . . .	4156
6.307 insert_no_store_hash_fn_imps.hpp File Reference . . . . .	4156
6.307.1 Detailed Description . . . . .	4156
6.308 insert_no_store_hash_fn_imps.hpp File Reference . . . . .	4156
6.308.1 Detailed Description . . . . .	4156
6.309 insert_store_hash_fn_imps.hpp File Reference . . . . .	4156
6.309.1 Detailed Description . . . . .	4156

6.310 <a href="#">insert_store_hash_fn_imps.hpp File Reference</a>	4156
6.310.1 Detailed Description	4156
6.311 <a href="#">invoke.h File Reference</a>	4156
6.311.1 Detailed Description	4157
6.312 <a href="#">iomanip File Reference</a>	4157
6.312.1 Detailed Description	4159
6.313 <a href="#">ios File Reference</a>	4159
6.313.1 Detailed Description	4159
6.314 <a href="#">ios_base.h File Reference</a>	4159
6.314.1 Detailed Description	4161
6.315 <a href="#">iosfwd File Reference</a>	4161
6.315.1 Detailed Description	4162
6.316 <a href="#">iostream File Reference</a>	4162
6.316.1 Detailed Description	4163
6.317 <a href="#">istream File Reference</a>	4163
6.317.1 Detailed Description	4164
6.318 <a href="#">istream.tcc File Reference</a>	4164
6.318.1 Detailed Description	4165
6.319 <a href="#">iterator File Reference</a>	4165
6.319.1 Detailed Description	4165
6.320 <a href="#">iterator File Reference</a>	4165
6.320.1 Detailed Description	4165
6.321 <a href="#">iterator File Reference</a>	4166
6.321.1 Detailed Description	4166
6.321.2 Function Documentation	4166
6.322 <a href="#">iterator.h File Reference</a>	4167
6.322.1 Detailed Description	4167
6.323 <a href="#">iterator.hpp File Reference</a>	4167
6.323.1 Detailed Description	4167
6.324 <a href="#">iterator_fn_imps.hpp File Reference</a>	4167
6.324.1 Detailed Description	4167
6.325 <a href="#">iterator_tracker.h File Reference</a>	4167
6.325.1 Detailed Description	4168
6.326 <a href="#">iterators_fn_imps.hpp File Reference</a>	4169
6.326.1 Detailed Description	4169
6.327 <a href="#">iterators_fn_imps.hpp File Reference</a>	4169
6.327.1 Detailed Description	4169
6.328 <a href="#">iterators_fn_imps.hpp File Reference</a>	4169
6.328.1 Detailed Description	4169

6.329 iterators_fn_imps.hpp File Reference . . . . .	4169
6.329.1 Detailed Description . . . . .	4169
6.330 iterators_fn_imps.hpp File Reference . . . . .	4169
6.330.1 Detailed Description . . . . .	4169
6.331 iterators_fn_imps.hpp File Reference . . . . .	4169
6.331.1 Detailed Description . . . . .	4169
6.332 iterators_fn_imps.hpp File Reference . . . . .	4170
6.332.1 Detailed Description . . . . .	4170
6.333 left_child_next_sibling_heap_.hpp File Reference . . . . .	4170
6.333.1 Detailed Description . . . . .	4170
6.334 lfts_config.h File Reference . . . . .	4170
6.334.1 Detailed Description . . . . .	4170
6.335 limits File Reference . . . . .	4171
6.335.1 Detailed Description . . . . .	4172
6.336 linear_probe_fn_imp.hpp File Reference . . . . .	4172
6.336.1 Detailed Description . . . . .	4172
6.337 list File Reference . . . . .	4172
6.337.1 Detailed Description . . . . .	4172
6.338 list File Reference . . . . .	4172
6.338.1 Detailed Description . . . . .	4173
6.339 list File Reference . . . . .	4173
6.339.1 Detailed Description . . . . .	4174
6.340 list File Reference . . . . .	4174
6.340.1 Detailed Description . . . . .	4174
6.341 list.tcc File Reference . . . . .	4175
6.341.1 Detailed Description . . . . .	4175
6.342 list_partition.h File Reference . . . . .	4175
6.342.1 Detailed Description . . . . .	4175
6.343 list_update_policy.hpp File Reference . . . . .	4175
6.343.1 Detailed Description . . . . .	4176
6.344 locale File Reference . . . . .	4176
6.344.1 Detailed Description . . . . .	4176
6.345 locale_classes.h File Reference . . . . .	4176
6.345.1 Detailed Description . . . . .	4176
6.346 locale_classes.tcc File Reference . . . . .	4176
6.346.1 Detailed Description . . . . .	4177
6.347 locale_conv.h File Reference . . . . .	4177
6.347.1 Detailed Description . . . . .	4178
6.348 locale_facets.h File Reference . . . . .	4178

6.348.1 Detailed Description	4179
6.349 locale_facets.tcc File Reference	4179
6.349.1 Detailed Description	4180
6.350 locale_facets_nonio.h File Reference	4180
6.350.1 Detailed Description	4180
6.351 locale_facets_nonio.tcc File Reference	4181
6.351.1 Detailed Description	4181
6.352 localefwd.h File Reference	4181
6.352.1 Detailed Description	4182
6.353 losertree.h File Reference	4182
6.353.1 Detailed Description	4183
6.354 lu_counter_metadata.hpp File Reference	4183
6.354.1 Detailed Description	4183
6.355 lu_map_.hpp File Reference	4183
6.355.1 Detailed Description	4184
6.356 macros.h File Reference	4184
6.356.1 Detailed Description	4185
6.356.2 Macro Definition Documentation	4185
6.357 malloc_allocator.h File Reference	4189
6.357.1 Detailed Description	4189
6.358 map File Reference	4189
6.358.1 Detailed Description	4189
6.359 map File Reference	4189
6.359.1 Detailed Description	4189
6.360 map File Reference	4190
6.360.1 Detailed Description	4190
6.361 map File Reference	4190
6.361.1 Detailed Description	4190
6.362 map.h File Reference	4191
6.362.1 Detailed Description	4191
6.363 map.h File Reference	4191
6.363.1 Detailed Description	4192
6.364 mask_array.h File Reference	4192
6.364.1 Detailed Description	4193
6.365 mask_based_range_hashing.hpp File Reference	4193
6.365.1 Detailed Description	4193
6.366 math.h File Reference	4193
6.366.1 Detailed Description	4193
6.367 memory File Reference	4193

---

6.367.1 Detailed Description . . . . .	4194
6.368 memory File Reference . . . . .	4194
6.368.1 Detailed Description . . . . .	4195
6.369 memory File Reference . . . . .	4195
6.369.1 Detailed Description . . . . .	4196
6.370 memory_resource File Reference . . . . .	4196
6.370.1 Detailed Description . . . . .	4196
6.371 memory_fwd.h File Reference . . . . .	4197
6.371.1 Detailed Description . . . . .	4197
6.372 merge.h File Reference . . . . .	4197
6.372.1 Detailed Description . . . . .	4197
6.373 messages_members.h File Reference . . . . .	4198
6.373.1 Detailed Description . . . . .	4198
6.374 mod_based_range_hashing.hpp File Reference . . . . .	4198
6.374.1 Detailed Description . . . . .	4198
6.375 move.h File Reference . . . . .	4198
6.375.1 Detailed Description . . . . .	4199
6.376 mt_allocator.h File Reference . . . . .	4199
6.376.1 Detailed Description . . . . .	4200
6.377 multimap.h File Reference . . . . .	4200
6.377.1 Detailed Description . . . . .	4201
6.378 multimap.h File Reference . . . . .	4201
6.378.1 Detailed Description . . . . .	4201
6.379 multiseq_selection.h File Reference . . . . .	4202
6.379.1 Detailed Description . . . . .	4202
6.380 multiset.h File Reference . . . . .	4202
6.380.1 Detailed Description . . . . .	4203
6.381 multiset.h File Reference . . . . .	4203
6.381.1 Detailed Description . . . . .	4204
6.382 multiway_merge.h File Reference . . . . .	4204
6.382.1 Detailed Description . . . . .	4207
6.382.2 Macro Definition Documentation . . . . .	4207
6.383 multiway_mergesort.h File Reference . . . . .	4207
6.383.1 Detailed Description . . . . .	4208
6.384 mutex File Reference . . . . .	4208
6.384.1 Detailed Description . . . . .	4209
6.385 nested_exception.h File Reference . . . . .	4209
6.385.1 Detailed Description . . . . .	4209
6.386 new File Reference . . . . .	4209



6.386.1 Detailed Description . . . . .	4210
6.386.2 Function Documentation . . . . .	4210
6.387 new_allocator.h File Reference . . . . .	4215
6.387.1 Detailed Description . . . . .	4215
6.388 node.hpp File Reference . . . . .	4215
6.388.1 Detailed Description . . . . .	4215
6.389 node.hpp File Reference . . . . .	4215
6.389.1 Detailed Description . . . . .	4216
6.390 node.hpp File Reference . . . . .	4216
6.390.1 Detailed Description . . . . .	4216
6.391 node_handle.h File Reference . . . . .	4216
6.391.1 Detailed Description . . . . .	4216
6.392 node_iterators.hpp File Reference . . . . .	4216
6.392.1 Detailed Description . . . . .	4217
6.393 node_iterators.hpp File Reference . . . . .	4217
6.393.1 Detailed Description . . . . .	4217
6.394 node_metadata_selector.hpp File Reference . . . . .	4217
6.394.1 Detailed Description . . . . .	4218
6.395 node_metadata_selector.hpp File Reference . . . . .	4218
6.395.1 Detailed Description . . . . .	4218
6.396 null_node_metadata.hpp File Reference . . . . .	4218
6.396.1 Detailed Description . . . . .	4218
6.397 numeric File Reference . . . . .	4218
6.397.1 Detailed Description . . . . .	4219
6.398 numeric File Reference . . . . .	4219
6.398.1 Detailed Description . . . . .	4219
6.399 numeric File Reference . . . . .	4220
6.399.1 Detailed Description . . . . .	4221
6.400 numeric File Reference . . . . .	4222
6.400.1 Detailed Description . . . . .	4222
6.400.2 Function Documentation . . . . .	4222
6.401 numeric_traits.h File Reference . . . . .	4223
6.401.1 Detailed Description . . . . .	4223
6.402 numeric_fwd.h File Reference . . . . .	4223
6.402.1 Detailed Description . . . . .	4225
6.403 omp_loop.h File Reference . . . . .	4225
6.403.1 Detailed Description . . . . .	4225
6.404 omp_loop_static.h File Reference . . . . .	4225
6.404.1 Detailed Description . . . . .	4225

6.405 <a href="#">opt_random.h</a> File Reference	4226
6.405.1 Detailed Description	4226
6.406 <a href="#">optional</a> File Reference	4226
6.406.1 Detailed Description	4229
6.406.2 Function Documentation	4229
6.407 <a href="#">order_statistics_imp.hpp</a> File Reference	4229
6.407.1 Detailed Description	4229
6.408 <a href="#">order_statistics_imp.hpp</a> File Reference	4229
6.408.1 Detailed Description	4229
6.409 <a href="#">ordered_base.h</a> File Reference	4229
6.409.1 Detailed Description	4230
6.410 <a href="#">os_defines.h</a> File Reference	4230
6.410.1 Detailed Description	4230
6.411 <a href="#">ostream</a> File Reference	4230
6.411.1 Detailed Description	4231
6.412 <a href="#">ostream.tcc</a> File Reference	4232
6.412.1 Detailed Description	4232
6.413 <a href="#">ostream_insert.h</a> File Reference	4232
6.413.1 Detailed Description	4232
6.414 <a href="#">ov_tree_map_.hpp</a> File Reference	4233
6.414.1 Detailed Description	4233
6.415 <a href="#">pairing_heap_.hpp</a> File Reference	4233
6.415.1 Detailed Description	4233
6.416 <a href="#">par_loop.h</a> File Reference	4234
6.416.1 Detailed Description	4234
6.417 <a href="#">parallel.h</a> File Reference	4234
6.417.1 Detailed Description	4234
6.418 <a href="#">parse_numbers.h</a> File Reference	4234
6.418.1 Detailed Description	4234
6.419 <a href="#">partial_sum.h</a> File Reference	4235
6.419.1 Detailed Description	4235
6.420 <a href="#">partition.h</a> File Reference	4235
6.420.1 Detailed Description	4236
6.420.2 Macro Definition Documentation	4236
6.421 <a href="#">pat_trie_.hpp</a> File Reference	4236
6.421.1 Detailed Description	4236
6.422 <a href="#">pat_trie_base.hpp</a> File Reference	4237
6.422.1 Detailed Description	4237
6.423 <a href="#">pod_char_traits.h</a> File Reference	4237

6.423.1 Detailed Description . . . . .	4238
6.424 point_const_iterator.hpp File Reference . . . . .	4238
6.424.1 Detailed Description . . . . .	4238
6.425 point_const_iterator.hpp File Reference . . . . .	4238
6.425.1 Detailed Description . . . . .	4239
6.426 point_const_iterator.hpp File Reference . . . . .	4239
6.426.1 Detailed Description . . . . .	4239
6.427 point_iterator.hpp File Reference . . . . .	4239
6.427.1 Detailed Description . . . . .	4239
6.428 point_iterators.hpp File Reference . . . . .	4239
6.428.1 Detailed Description . . . . .	4240
6.429 pointer.h File Reference . . . . .	4240
6.429.1 Detailed Description . . . . .	4242
6.430 policy_access_fn_imps.hpp File Reference . . . . .	4242
6.430.1 Detailed Description . . . . .	4242
6.431 policy_access_fn_imps.hpp File Reference . . . . .	4242
6.431.1 Detailed Description . . . . .	4242
6.432 policy_access_fn_imps.hpp File Reference . . . . .	4242
6.432.1 Detailed Description . . . . .	4242
6.433 policy_access_fn_imps.hpp File Reference . . . . .	4242
6.433.1 Detailed Description . . . . .	4242
6.434 policy_access_fn_imps.hpp File Reference . . . . .	4242
6.434.1 Detailed Description . . . . .	4242
6.435 policy_access_fn_imps.hpp File Reference . . . . .	4243
6.435.1 Detailed Description . . . . .	4243
6.436 policy_access_fn_imps.hpp File Reference . . . . .	4243
6.436.1 Detailed Description . . . . .	4243
6.437 pool_allocator.h File Reference . . . . .	4243
6.437.1 Detailed Description . . . . .	4243
6.438 postypes.h File Reference . . . . .	4243
6.438.1 Detailed Description . . . . .	4244
6.439 predefined_ops.h File Reference . . . . .	4244
6.439.1 Detailed Description . . . . .	4245
6.440 prefix_search_node_update_imp.hpp File Reference . . . . .	4245
6.440.1 Detailed Description . . . . .	4245
6.441 priority_queue.hpp File Reference . . . . .	4245
6.441.1 Detailed Description . . . . .	4246
6.442 priority_queue_base_dispatch.hpp File Reference . . . . .	4246
6.442.1 Detailed Description . . . . .	4246

---

6.443 probe_fn_base.hpp File Reference . . . . .	4246
6.443.1 Detailed Description . . . . .	4246
6.444 profiler.h File Reference . . . . .	4247
6.444.1 Detailed Description . . . . .	4249
6.445 profiler_algos.h File Reference . . . . .	4249
6.445.1 Detailed Description . . . . .	4249
6.446 profiler_container_size.h File Reference . . . . .	4249
6.446.1 Detailed Description . . . . .	4250
6.447 profiler_hash_func.h File Reference . . . . .	4250
6.447.1 Detailed Description . . . . .	4250
6.448 profiler_hashtable_size.h File Reference . . . . .	4250
6.448.1 Detailed Description . . . . .	4251
6.449 profiler_list_to_slist.h File Reference . . . . .	4251
6.449.1 Detailed Description . . . . .	4251
6.450 profiler_list_to_vector.h File Reference . . . . .	4251
6.450.1 Detailed Description . . . . .	4252
6.451 profiler_map_to_unordered_map.h File Reference . . . . .	4252
6.451.1 Detailed Description . . . . .	4252
6.452 profiler_node.h File Reference . . . . .	4252
6.452.1 Detailed Description . . . . .	4253
6.453 profiler_state.h File Reference . . . . .	4253
6.453.1 Detailed Description . . . . .	4253
6.454 profiler_trace.h File Reference . . . . .	4254
6.454.1 Detailed Description . . . . .	4256
6.455 profiler_vector_size.h File Reference . . . . .	4256
6.455.1 Detailed Description . . . . .	4256
6.456 profiler_vector_to_list.h File Reference . . . . .	4256
6.456.1 Detailed Description . . . . .	4257
6.457 propagate_const File Reference . . . . .	4257
6.457.1 Detailed Description . . . . .	4259
6.458 ptr_traits.h File Reference . . . . .	4259
6.458.1 Detailed Description . . . . .	4260
6.459 quadratic_probe_fn_imp.hpp File Reference . . . . .	4260
6.459.1 Detailed Description . . . . .	4260
6.460 queue File Reference . . . . .	4260
6.460.1 Detailed Description . . . . .	4260
6.461 queue.h File Reference . . . . .	4260
6.461.1 Detailed Description . . . . .	4261
6.461.2 Macro Definition Documentation . . . . .	4261

6.462 quicksort.h File Reference . . . . .	4261
6.462.1 Detailed Description . . . . .	4261
6.463 quoted_string.h File Reference . . . . .	4261
6.463.1 Detailed Description . . . . .	4262
6.464 r_erase_fn_imps.hpp File Reference . . . . .	4262
6.464.1 Detailed Description . . . . .	4262
6.465 r_erase_fn_imps.hpp File Reference . . . . .	4262
6.465.1 Detailed Description . . . . .	4262
6.466 random File Reference . . . . .	4262
6.466.1 Detailed Description . . . . .	4262
6.467 random File Reference . . . . .	4263
6.467.1 Detailed Description . . . . .	4263
6.468 random.h File Reference . . . . .	4263
6.468.1 Detailed Description . . . . .	4267
6.469 random.tcc File Reference . . . . .	4267
6.469.1 Detailed Description . . . . .	4271
6.470 random.tcc File Reference . . . . .	4271
6.470.1 Detailed Description . . . . .	4273
6.471 random_number.h File Reference . . . . .	4273
6.471.1 Detailed Description . . . . .	4273
6.472 random_shuffle.h File Reference . . . . .	4274
6.472.1 Detailed Description . . . . .	4274
6.473 range_access.h File Reference . . . . .	4274
6.473.1 Detailed Description . . . . .	4276
6.474 ranged_hash_fn.hpp File Reference . . . . .	4276
6.474.1 Detailed Description . . . . .	4276
6.475 ranged_probe_fn.hpp File Reference . . . . .	4276
6.475.1 Detailed Description . . . . .	4277
6.476 ratio File Reference . . . . .	4277
6.476.1 Detailed Description . . . . .	4277
6.477 ratio File Reference . . . . .	4278
6.477.1 Detailed Description . . . . .	4278
6.478 ratio File Reference . . . . .	4278
6.478.1 Detailed Description . . . . .	4278
6.479 rb_tree File Reference . . . . .	4278
6.479.1 Detailed Description . . . . .	4279
6.480 rb_tree.hpp File Reference . . . . .	4279
6.480.1 Detailed Description . . . . .	4279
6.481 rc.hpp File Reference . . . . .	4279

---

6.481.1 Detailed Description . . . . .	4280
6.482 rc_binomial_heap.hpp File Reference . . . . .	4280
6.482.1 Detailed Description . . . . .	4280
6.483 rc_string_base.h File Reference . . . . .	4280
6.483.1 Detailed Description . . . . .	4280
6.484 refwrap.h File Reference . . . . .	4281
6.484.1 Detailed Description . . . . .	4281
6.485 regex File Reference . . . . .	4282
6.485.1 Detailed Description . . . . .	4282
6.486 regex File Reference . . . . .	4282
6.486.1 Detailed Description . . . . .	4282
6.487 regex.h File Reference . . . . .	4282
6.487.1 Detailed Description . . . . .	4287
6.488 regex.tcc File Reference . . . . .	4287
6.488.1 Detailed Description . . . . .	4288
6.489 regex_automaton.h File Reference . . . . .	4288
6.489.1 Detailed Description . . . . .	4289
6.490 regex_automaton.tcc File Reference . . . . .	4289
6.490.1 Detailed Description . . . . .	4289
6.491 regex_compiler.h File Reference . . . . .	4289
6.491.1 Detailed Description . . . . .	4290
6.492 regex_compiler.tcc File Reference . . . . .	4290
6.492.1 Detailed Description . . . . .	4290
6.493 regex_constants.h File Reference . . . . .	4290
6.493.1 Detailed Description . . . . .	4292
6.494 regex_error.h File Reference . . . . .	4292
6.494.1 Detailed Description . . . . .	4293
6.495 regex_executor.h File Reference . . . . .	4293
6.495.1 Detailed Description . . . . .	4293
6.496 regex_executor.tcc File Reference . . . . .	4293
6.496.1 Detailed Description . . . . .	4293
6.497 regex_scanner.h File Reference . . . . .	4293
6.497.1 Detailed Description . . . . .	4294
6.498 regex_scanner.tcc File Reference . . . . .	4294
6.498.1 Detailed Description . . . . .	4294
6.499 resize_fn_imps.hpp File Reference . . . . .	4294
6.499.1 Detailed Description . . . . .	4294
6.500 resize_fn_imps.hpp File Reference . . . . .	4294
6.500.1 Detailed Description . . . . .	4294

6.501 <a href="#">resize_no_store_hash_fn_imps.hpp</a> File Reference	4294
6.501.1 Detailed Description	4294
6.502 <a href="#">resize_no_store_hash_fn_imps.hpp</a> File Reference	4295
6.502.1 Detailed Description	4295
6.503 <a href="#">resize_policy.hpp</a> File Reference	4295
6.503.1 Detailed Description	4295
6.504 <a href="#">resize_store_hash_fn_imps.hpp</a> File Reference	4295
6.504.1 Detailed Description	4295
6.505 <a href="#">resize_store_hash_fn_imps.hpp</a> File Reference	4295
6.505.1 Detailed Description	4295
6.506 <a href="#">rope</a> File Reference	4295
6.506.1 Detailed Description	4299
6.507 <a href="#">ropeimpl.h</a> File Reference	4299
6.507.1 Detailed Description	4299
6.508 <a href="#">rotate_fn_imps.hpp</a> File Reference	4299
6.508.1 Detailed Description	4299
6.509 <a href="#">rotate_fn_imps.hpp</a> File Reference	4299
6.509.1 Detailed Description	4299
6.510 <a href="#">safe_base.h</a> File Reference	4300
6.510.1 Detailed Description	4300
6.511 <a href="#">safe_container.h</a> File Reference	4300
6.511.1 Detailed Description	4300
6.512 <a href="#">safe_iterator.h</a> File Reference	4300
6.512.1 Detailed Description	4302
6.513 <a href="#">safe_iterator.tcc</a> File Reference	4302
6.513.1 Detailed Description	4302
6.514 <a href="#">safe_local_iterator.h</a> File Reference	4303
6.514.1 Detailed Description	4303
6.515 <a href="#">safe_local_iterator.tcc</a> File Reference	4303
6.515.1 Detailed Description	4304
6.516 <a href="#">safe_sequence.h</a> File Reference	4304
6.516.1 Detailed Description	4304
6.517 <a href="#">safe_sequence.tcc</a> File Reference	4304
6.517.1 Detailed Description	4304
6.518 <a href="#">safe_unordered_base.h</a> File Reference	4305
6.518.1 Detailed Description	4305
6.519 <a href="#">safe_unordered_container.h</a> File Reference	4305
6.519.1 Detailed Description	4305
6.520 <a href="#">safe_unordered_container.tcc</a> File Reference	4305

6.520.1 Detailed Description . . . . .	4305
6.521 sample_probe_fn.hpp File Reference . . . . .	4306
6.521.1 Detailed Description . . . . .	4306
6.522 sample_range_hashing.hpp File Reference . . . . .	4306
6.522.1 Detailed Description . . . . .	4306
6.523 sample_ranged_hash_fn.hpp File Reference . . . . .	4306
6.523.1 Detailed Description . . . . .	4306
6.524 sample_ranged_probe_fn.hpp File Reference . . . . .	4307
6.524.1 Detailed Description . . . . .	4307
6.525 sample_resize_policy.hpp File Reference . . . . .	4307
6.525.1 Detailed Description . . . . .	4307
6.526 sample_resize_trigger.hpp File Reference . . . . .	4307
6.526.1 Detailed Description . . . . .	4307
6.527 sample_size_policy.hpp File Reference . . . . .	4308
6.527.1 Detailed Description . . . . .	4308
6.528 sample_tree_node_update.hpp File Reference . . . . .	4308
6.528.1 Detailed Description . . . . .	4308
6.529 sample_trie_access_traits.hpp File Reference . . . . .	4308
6.529.1 Detailed Description . . . . .	4308
6.530 sample_trie_node_update.hpp File Reference . . . . .	4309
6.530.1 Detailed Description . . . . .	4309
6.531 sample_update_policy.hpp File Reference . . . . .	4309
6.531.1 Detailed Description . . . . .	4309
6.532 scoped_allocator File Reference . . . . .	4309
6.532.1 Detailed Description . . . . .	4310
6.533 search.h File Reference . . . . .	4310
6.533.1 Detailed Description . . . . .	4310
6.534 set File Reference . . . . .	4310
6.534.1 Detailed Description . . . . .	4311
6.535 set File Reference . . . . .	4311
6.535.1 Detailed Description . . . . .	4311
6.536 set File Reference . . . . .	4311
6.536.1 Detailed Description . . . . .	4311
6.537 set File Reference . . . . .	4311
6.537.1 Detailed Description . . . . .	4312
6.538 set.h File Reference . . . . .	4312
6.538.1 Detailed Description . . . . .	4313
6.539 set.h File Reference . . . . .	4313
6.539.1 Detailed Description . . . . .	4313



6.540 set_operations.h File Reference . . . . .	4313
6.540.1 Detailed Description . . . . .	4314
6.541 settings.h File Reference . . . . .	4314
6.541.1 Detailed Description . . . . .	4314
6.541.2 parallelization_decision . . . . .	4315
6.541.3 Macro Definition Documentation . . . . .	4315
6.542 shared_mutex File Reference . . . . .	4316
6.542.1 Detailed Description . . . . .	4316
6.543 shared_ptr.h File Reference . . . . .	4316
6.543.1 Detailed Description . . . . .	4318
6.544 shared_ptr.h File Reference . . . . .	4318
6.544.1 Detailed Description . . . . .	4320
6.544.2 Function Documentation . . . . .	4320
6.545 shared_ptr_atomic.h File Reference . . . . .	4321
6.545.1 Detailed Description . . . . .	4322
6.546 shared_ptr_base.h File Reference . . . . .	4323
6.546.1 Detailed Description . . . . .	4324
6.547 size_fn_imps.hpp File Reference . . . . .	4324
6.547.1 Detailed Description . . . . .	4324
6.548 slice_array.h File Reference . . . . .	4324
6.548.1 Detailed Description . . . . .	4325
6.549 slist File Reference . . . . .	4325
6.549.1 Detailed Description . . . . .	4326
6.550 sort.h File Reference . . . . .	4326
6.550.1 Detailed Description . . . . .	4326
6.551 specfun.h File Reference . . . . .	4326
6.551.1 Detailed Description . . . . .	4329
6.552 splay_fn_imps.hpp File Reference . . . . .	4329
6.552.1 Detailed Description . . . . .	4329
6.553 splay_tree_.hpp File Reference . . . . .	4329
6.553.1 Detailed Description . . . . .	4329
6.554 split_fn_imps.hpp File Reference . . . . .	4330
6.554.1 Detailed Description . . . . .	4330
6.555 split_join_fn_imps.hpp File Reference . . . . .	4330
6.555.1 Detailed Description . . . . .	4330
6.556 split_join_fn_imps.hpp File Reference . . . . .	4330
6.556.1 Detailed Description . . . . .	4330
6.557 split_join_fn_imps.hpp File Reference . . . . .	4330
6.557.1 Detailed Description . . . . .	4330

6.558 <a href="#">split_join_fn_imps.hpp</a> File Reference	4330
6.558.1 Detailed Description	4330
6.559 <a href="#">split_join_fn_imps.hpp</a> File Reference	4330
6.559.1 Detailed Description	4330
6.560 <a href="#">split_join_fn_imps.hpp</a> File Reference	4331
6.560.1 Detailed Description	4331
6.561 <a href="#">split_join_fn_imps.hpp</a> File Reference	4331
6.561.1 Detailed Description	4331
6.562 <a href="#">split_join_fn_imps.hpp</a> File Reference	4331
6.562.1 Detailed Description	4331
6.563 <a href="#">split_join_fn_imps.hpp</a> File Reference	4331
6.563.1 Detailed Description	4331
6.564 <a href="#">sso_string_base.h</a> File Reference	4331
6.564.1 Detailed Description	4331
6.565 <a href="#">sstream</a> File Reference	4331
6.565.1 Detailed Description	4332
6.566 <a href="#">sstream.tcc</a> File Reference	4332
6.566.1 Detailed Description	4332
6.567 <a href="#">stack</a> File Reference	4333
6.567.1 Detailed Description	4333
6.568 <a href="#">standard_policies.hpp</a> File Reference	4333
6.568.1 Detailed Description	4333
6.568.2 Enumeration Type Documentation	4333
6.569 <a href="#">std_abs.h</a> File Reference	4334
6.569.1 Detailed Description	4334
6.570 <a href="#">std_function.h</a> File Reference	4334
6.570.1 Detailed Description	4335
6.571 <a href="#">std_mutex.h</a> File Reference	4335
6.571.1 Detailed Description	4336
6.572 <a href="#">stdc++.h</a> File Reference	4336
6.572.1 Detailed Description	4336
6.573 <a href="#">stdexcept</a> File Reference	4336
6.573.1 Detailed Description	4336
6.574 <a href="#">stdio_filebuf.h</a> File Reference	4337
6.574.1 Detailed Description	4337
6.575 <a href="#">stdio_sync_filebuf.h</a> File Reference	4337
6.575.1 Detailed Description	4337
6.576 <a href="#">stdlib.h</a> File Reference	4337
6.576.1 Detailed Description	4337

6.577 stdtr1c++.h File Reference . . . . .	4337
6.577.1 Detailed Description . . . . .	4337
6.578 stl_algo.h File Reference . . . . .	4338
6.578.1 Detailed Description . . . . .	4347
6.578.2 Function Documentation . . . . .	4347
6.579 stl_algobase.h File Reference . . . . .	4348
6.579.1 Detailed Description . . . . .	4351
6.580 stl_bvector.h File Reference . . . . .	4351
6.580.1 Detailed Description . . . . .	4352
6.581 stl_construct.h File Reference . . . . .	4352
6.581.1 Detailed Description . . . . .	4352
6.582 stl_deque.h File Reference . . . . .	4353
6.582.1 Detailed Description . . . . .	4355
6.582.2 Macro Definition Documentation . . . . .	4355
6.583 stl_function.h File Reference . . . . .	4355
6.583.1 Detailed Description . . . . .	4357
6.584 stl_heap.h File Reference . . . . .	4358
6.584.1 Detailed Description . . . . .	4359
6.585 stl_iterator.h File Reference . . . . .	4359
6.585.1 Detailed Description . . . . .	4362
6.586 stl_iterator.h File Reference . . . . .	4363
6.586.1 Detailed Description . . . . .	4363
6.587 stl_iterator_base_funcs.h File Reference . . . . .	4363
6.587.1 Detailed Description . . . . .	4364
6.588 stl_iterator_base_types.h File Reference . . . . .	4364
6.588.1 Detailed Description . . . . .	4365
6.589 stl_list.h File Reference . . . . .	4365
6.589.1 Detailed Description . . . . .	4366
6.590 stl_map.h File Reference . . . . .	4366
6.590.1 Detailed Description . . . . .	4367
6.591 stl_multimap.h File Reference . . . . .	4367
6.591.1 Detailed Description . . . . .	4368
6.592 stl_multiset.h File Reference . . . . .	4368
6.592.1 Detailed Description . . . . .	4369
6.593 stl_numeric.h File Reference . . . . .	4369
6.593.1 Detailed Description . . . . .	4370
6.594 stl_pair.h File Reference . . . . .	4370
6.594.1 Detailed Description . . . . .	4371
6.595 stl_queue.h File Reference . . . . .	4371

6.595.1 Detailed Description	4371
6.596 <a href="#">stl_raw_storage_iter.h</a> File Reference	4372
6.596.1 Detailed Description	4372
6.597 <a href="#">stl_relops.h</a> File Reference	4372
6.597.1 Detailed Description	4372
6.598 <a href="#">stl_set.h</a> File Reference	4373
6.598.1 Detailed Description	4373
6.599 <a href="#">stl_stack.h</a> File Reference	4373
6.599.1 Detailed Description	4374
6.600 <a href="#">stl_tempbuf.h</a> File Reference	4374
6.600.1 Detailed Description	4374
6.601 <a href="#">stl_tree.h</a> File Reference	4375
6.601.1 Detailed Description	4376
6.602 <a href="#">stl_uninitialized.h</a> File Reference	4376
6.602.1 Detailed Description	4377
6.603 <a href="#">stl_vector.h</a> File Reference	4377
6.603.1 Detailed Description	4378
6.604 <a href="#">stream_iterator.h</a> File Reference	4378
6.604.1 Detailed Description	4379
6.605 <a href="#">streambuf</a> File Reference	4379
6.605.1 Detailed Description	4379
6.606 <a href="#">streambuf.tcc</a> File Reference	4380
6.606.1 Detailed Description	4380
6.607 <a href="#">streambuf_iterator.h</a> File Reference	4380
6.607.1 Detailed Description	4381
6.608 <a href="#">string</a> File Reference	4381
6.608.1 Detailed Description	4381
6.609 <a href="#">string</a> File Reference	4381
6.609.1 Detailed Description	4384
6.610 <a href="#">string</a> File Reference	4384
6.610.1 Detailed Description	4384
6.611 <a href="#">string_conversions.h</a> File Reference	4384
6.611.1 Detailed Description	4385
6.612 <a href="#">string_view</a> File Reference	4385
6.612.1 Detailed Description	4386
6.613 <a href="#">string_view.tcc</a> File Reference	4387
6.613.1 Detailed Description	4387
6.614 <a href="#">string_view.tcc</a> File Reference	4387
6.614.1 Detailed Description	4387

6.615 stringfwd.h File Reference . . . . .	4387
6.615.1 Detailed Description . . . . .	4388
6.616 stringstream File Reference . . . . .	4388
6.616.1 Detailed Description . . . . .	4388
6.617 synth_access_traits.hpp File Reference . . . . .	4388
6.617.1 Detailed Description . . . . .	4388
6.618 system_error File Reference . . . . .	4388
6.618.1 Detailed Description . . . . .	4389
6.619 system_error File Reference . . . . .	4389
6.619.1 Detailed Description . . . . .	4390
6.620 tag_and_trait.hpp File Reference . . . . .	4390
6.620.1 Detailed Description . . . . .	4391
6.621 tags.h File Reference . . . . .	4391
6.621.1 Detailed Description . . . . .	4392
6.622 tgmth.h File Reference . . . . .	4392
6.622.1 Detailed Description . . . . .	4392
6.623 thin_heap_.hpp File Reference . . . . .	4392
6.623.1 Detailed Description . . . . .	4393
6.624 thread File Reference . . . . .	4393
6.624.1 Detailed Description . . . . .	4394
6.625 throw_allocator.h File Reference . . . . .	4394
6.625.1 Detailed Description . . . . .	4395
6.626 time_members.h File Reference . . . . .	4395
6.626.1 Detailed Description . . . . .	4395
6.627 trace_fn_imps.hpp File Reference . . . . .	4395
6.627.1 Detailed Description . . . . .	4395
6.628 trace_fn_imps.hpp File Reference . . . . .	4395
6.628.1 Detailed Description . . . . .	4395
6.629 trace_fn_imps.hpp File Reference . . . . .	4396
6.629.1 Detailed Description . . . . .	4396
6.630 trace_fn_imps.hpp File Reference . . . . .	4396
6.630.1 Detailed Description . . . . .	4396
6.631 trace_fn_imps.hpp File Reference . . . . .	4396
6.631.1 Detailed Description . . . . .	4396
6.632 trace_fn_imps.hpp File Reference . . . . .	4396
6.632.1 Detailed Description . . . . .	4396
6.633 trace_fn_imps.hpp File Reference . . . . .	4396
6.633.1 Detailed Description . . . . .	4396
6.634 trace_fn_imps.hpp File Reference . . . . .	4396

---

6.634.1 Detailed Description . . . . .	4396
6.635 traits.hpp File Reference . . . . .	4397
6.635.1 Detailed Description . . . . .	4397
6.636 traits.hpp File Reference . . . . .	4397
6.636.1 Detailed Description . . . . .	4397
6.637 traits.hpp File Reference . . . . .	4397
6.637.1 Detailed Description . . . . .	4398
6.638 traits.hpp File Reference . . . . .	4398
6.638.1 Detailed Description . . . . .	4398
6.639 traits.hpp File Reference . . . . .	4398
6.639.1 Detailed Description . . . . .	4398
6.640 traits.hpp File Reference . . . . .	4399
6.640.1 Detailed Description . . . . .	4399
6.641 tree_policy.hpp File Reference . . . . .	4399
6.641.1 Detailed Description . . . . .	4399
6.642 tree_trace_base.hpp File Reference . . . . .	4399
6.642.1 Detailed Description . . . . .	4399
6.643 trie_policy.hpp File Reference . . . . .	4400
6.643.1 Detailed Description . . . . .	4400
6.644 trie_policy_base.hpp File Reference . . . . .	4400
6.644.1 Detailed Description . . . . .	4400
6.645 trie_string_access_traits_imp.hpp File Reference . . . . .	4401
6.645.1 Detailed Description . . . . .	4401
6.646 tuple File Reference . . . . .	4401
6.646.1 Detailed Description . . . . .	4403
6.647 tuple File Reference . . . . .	4403
6.647.1 Detailed Description . . . . .	4403
6.648 type_traits File Reference . . . . .	4404
6.648.1 Detailed Description . . . . .	4408
6.648.2 Macro Definition Documentation . . . . .	4408
6.649 type_traits File Reference . . . . .	4408
6.649.1 Detailed Description . . . . .	4409
6.650 type_traits File Reference . . . . .	4409
6.650.1 Detailed Description . . . . .	4412
6.651 type_traits.h File Reference . . . . .	4412
6.651.1 Detailed Description . . . . .	4413
6.652 type_utils.hpp File Reference . . . . .	4413
6.652.1 Detailed Description . . . . .	4413
6.653 typeindex File Reference . . . . .	4413

6.653.1 Detailed Description . . . . .	4414
6.654 typeinfo File Reference . . . . .	4414
6.654.1 Detailed Description . . . . .	4414
6.655 typelist.h File Reference . . . . .	4414
6.655.1 Detailed Description . . . . .	4415
6.656 types.h File Reference . . . . .	4416
6.656.1 Detailed Description . . . . .	4416
6.657 types_traits.hpp File Reference . . . . .	4416
6.657.1 Detailed Description . . . . .	4417
6.658 uniform_int_dist.h File Reference . . . . .	4417
6.658.1 Detailed Description . . . . .	4417
6.659 unique_copy.h File Reference . . . . .	4417
6.659.1 Detailed Description . . . . .	4418
6.660 unique_ptr.h File Reference . . . . .	4418
6.660.1 Detailed Description . . . . .	4419
6.661 unordered_base.h File Reference . . . . .	4419
6.661.1 Detailed Description . . . . .	4420
6.662 unordered_map File Reference . . . . .	4420
6.662.1 Detailed Description . . . . .	4420
6.663 unordered_map File Reference . . . . .	4420
6.663.1 Detailed Description . . . . .	4421
6.664 unordered_map File Reference . . . . .	4421
6.664.1 Detailed Description . . . . .	4422
6.665 unordered_map File Reference . . . . .	4422
6.665.1 Detailed Description . . . . .	4422
6.666 unordered_map.h File Reference . . . . .	4422
6.666.1 Detailed Description . . . . .	4423
6.667 unordered_set File Reference . . . . .	4424
6.667.1 Detailed Description . . . . .	4424
6.668 unordered_set File Reference . . . . .	4424
6.668.1 Detailed Description . . . . .	4425
6.669 unordered_set File Reference . . . . .	4425
6.669.1 Detailed Description . . . . .	4426
6.670 unordered_set File Reference . . . . .	4426
6.670.1 Detailed Description . . . . .	4426
6.671 unordered_set.h File Reference . . . . .	4426
6.671.1 Detailed Description . . . . .	4427
6.672 update_fn_imps.hpp File Reference . . . . .	4427
6.672.1 Detailed Description . . . . .	4427

6.673 utility File Reference . . . . .	4428
6.673.1 Detailed Description . . . . .	4429
6.674 utility File Reference . . . . .	4429
6.674.1 Detailed Description . . . . .	4430
6.675 valarray File Reference . . . . .	4430
6.675.1 Detailed Description . . . . .	4434
6.676 valarray_after.h File Reference . . . . .	4434
6.676.1 Detailed Description . . . . .	4443
6.677 valarray_array.h File Reference . . . . .	4443
6.677.1 Detailed Description . . . . .	4451
6.678 valarray_array.tcc File Reference . . . . .	4451
6.678.1 Detailed Description . . . . .	4452
6.679 valarray_before.h File Reference . . . . .	4452
6.679.1 Detailed Description . . . . .	4452
6.680 vector File Reference . . . . .	4453
6.680.1 Detailed Description . . . . .	4453
6.681 vector File Reference . . . . .	4453
6.681.1 Detailed Description . . . . .	4454
6.682 vector File Reference . . . . .	4454
6.682.1 Detailed Description . . . . .	4454
6.683 vector File Reference . . . . .	4454
6.683.1 Detailed Description . . . . .	4455
6.684 vector.tcc File Reference . . . . .	4455
6.684.1 Detailed Description . . . . .	4455
6.685 vstring.h File Reference . . . . .	4455
6.685.1 Detailed Description . . . . .	4458
6.686 vstring.tcc File Reference . . . . .	4458
6.686.1 Detailed Description . . . . .	4459
6.687 vstring_fwd.h File Reference . . . . .	4459
6.687.1 Detailed Description . . . . .	4459
6.688 vstring_util.h File Reference . . . . .	4459
6.688.1 Detailed Description . . . . .	4460
6.689 workstealing.h File Reference . . . . .	4460
6.689.1 Detailed Description . . . . .	4460
<b>Index</b>	<b>4461</b>

## 1 Mathematical Special Functions



## 1.1 Introduction and History

The first significant library upgrade on the road to C++2011, [TR1](#), included a set of 23 mathematical functions that significantly extended the standard transcendental functions inherited from C and declared in `<cmath>`.

Although most components from TR1 were eventually adopted for C++11 these math functions were left behind out of concern for implementability. The math functions were published as a separate international standard [IS 29124 - Extensions to the C++ Library to Support Mathematical Special Functions](#).

For C++17 these functions were incorporated into the main standard.

## 1.2 Contents

The following functions are implemented in namespace `std`:

- [assoc\\_laguerre](#) - Associated Laguerre functions
- [assoc\\_legendre](#) - Associated Legendre functions
- [beta](#) - Beta functions
- [comp\\_ellint\\_1](#) - Complete elliptic functions of the first kind
- [comp\\_ellint\\_2](#) - Complete elliptic functions of the second kind
- [comp\\_ellint\\_3](#) - Complete elliptic functions of the third kind
- [cyl\\_bessel\\_i](#) - Regular modified cylindrical Bessel functions
- [cyl\\_bessel\\_j](#) - Cylindrical Bessel functions of the first kind
- [cyl\\_bessel\\_k](#) - Irregular modified cylindrical Bessel functions
- [cyl\\_neumann](#) - Cylindrical Neumann functions or Cylindrical Bessel functions of the second kind
- [ellint\\_1](#) - Incomplete elliptic functions of the first kind
- [ellint\\_2](#) - Incomplete elliptic functions of the second kind
- [ellint\\_3](#) - Incomplete elliptic functions of the third kind
- [expint](#) - The exponential integral
- [hermite](#) - Hermite polynomials
- [laguerre](#) - Laguerre functions
- [legendre](#) - Legendre polynomials
- [riemann\\_zeta](#) - The Riemann zeta function
- [sph\\_bessel](#) - Spherical Bessel functions
- [sph\\_legendre](#) - Spherical Legendre functions
- [sph\\_neumann](#) - Spherical Neumann functions

The hypergeometric functions were stricken from the TR29124 and C++17 versions of this math library because of implementation concerns. However, since they were in the TR1 version and since they are popular we kept them as an extension in namespace `__gnu_cxx`:

- [conf\\_hyperg](#) - Confluent hypergeometric functions
- [hyperg](#) - Hypergeometric functions

## 1.3 General Features

### 1.3.1 Argument Promotion

The arguments supplied to the non-suffixed functions will be promoted according to the following rules: 1. If any argument intended to be floating point is given an integral value That integral value is promoted to double. 2. All floating point arguments are promoted up to the largest floating point precision among them.

### 1.3.2 NaN Arguments

If any of the floating point arguments supplied to these functions is invalid or NaN (`std::numeric_limits<Tp>::quiet_NaN`), the value NaN is returned.

## 1.4 Implementation

We strive to implement the underlying math with type generic algorithms to the greatest extent possible. In practice, the functions are thin wrappers that dispatch to function templates. Type dependence is controlled with `std::numeric_limits` and functions thereof.

We don't promote `float` to `double` or `double` to `long double` reflexively. The goal is for `float` functions to operate more quickly, at the cost of `float` accuracy and possibly a smaller domain of validity. Similarly, `long double` should give you more dynamic range and slightly more precision than `double` on many systems.

## 1.5 Testing

These functions have been tested against equivalent implementations from the [Gnu Scientific Library](http://www.gnu.org/software/gsl/), [GSL](http://www.boost.org/doc/libs/1_60_0/libs/math/doc/html/index.html) and [Boost](http://www.boost.org/doc/libs/1_60_0/libs/math/doc/html/index.html) and the ratio

$$\frac{|f - f_{test}|}{|f_{test}|}$$

is generally found to be within  $10^{-15}$  for 64-bit double on linux-x86\_64 systems over most of the ranges of validity.

**Todo** Provide accuracy comparisons on a per-function basis for a small number of targets.

## 1.6 General Bibliography

See also

Abramowitz and Stegun: Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables Edited by Milton Abramowitz and Irene A. Stegun, National Bureau of Standards Applied Mathematics Series - 55 Issued June 1964, Tenth Printing, December 1972, with corrections Electronic versions of A&S abound including both pdf and navigable html.

for example <http://people.math.sfu.ca/~cbm/aands/>

The old A&S has been redone as the NIST Digital Library of Mathematical Functions: <http://dlmf.nist.gov/> This version is far more navigable and includes more recent work.

An Atlas of Functions: with Equator, the Atlas Function Calculator 2nd Edition, by Oldham, Keith B., Myland, Jan, Spanier, Jerome

Asymptotics and Special Functions by Frank W. J. Olver, Academic Press, 1974

Numerical Recipes in C, The Art of Scientific Computing, by William H. Press, Second Ed., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1992

The Special Functions and Their Approximations: Volumes 1 and 2, by Yudell L. Luke, Academic Press, 1969

## 2 Todo List

Member [\\_\\_gnu\\_cxx::distance](#) ([\\_InputIterator \\_\\_first](#), [\\_InputIterator \\_\\_last](#), [\\_Distance &\\_\\_n](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class [\\_\\_gnu\\_cxx::hash\\_map](#)<[\\_Key](#), [\\_Tp](#), [\\_HashFn](#), [\\_EqualKey](#), [\\_Alloc](#)>

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class [\\_\\_gnu\\_cxx::hash\\_multimap](#)<[\\_Key](#), [\\_Tp](#), [\\_HashFn](#), [\\_EqualKey](#), [\\_Alloc](#)>

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class [\\_\\_gnu\\_cxx::hash\\_multiset](#)<[\\_Value](#), [\\_HashFcn](#), [\\_EqualKey](#), [\\_Alloc](#)>

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class [\\_\\_gnu\\_cxx::hash\\_set](#)<[\\_Value](#), [\\_HashFcn](#), [\\_EqualKey](#), [\\_Alloc](#)>

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member [\\_\\_gnu\\_cxx::power](#) ([\\_Tp \\_\\_x](#), [\\_Integer \\_\\_n](#), [\\_MonoidOperation \\_\\_monoid\\_op](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member [\\_\\_gnu\\_cxx::power](#) ([\\_Tp \\_\\_x](#), [\\_Integer \\_\\_n](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member [\\_\\_gnu\\_cxx::random\\_sample](#) ([\\_InputIterator \\_\\_first](#), [\\_InputIterator \\_\\_last](#), [\\_RandomAccessIterator \\_\\_↵out\\_first](#), [\\_RandomAccessIterator \\_\\_out\\_last](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member [\\_\\_gnu\\_cxx::random\\_sample](#) ([\\_InputIterator \\_\\_first](#), [\\_InputIterator \\_\\_last](#), [\\_RandomAccessIterator \\_\\_↵out\\_first](#), [\\_RandomAccessIterator \\_\\_out\\_last](#), [\\_RandomNumberGenerator &\\_\\_rand](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member [\\_\\_gnu\\_cxx::random\\_sample\\_n](#) ([\\_ForwardIterator \\_\\_first](#), [\\_ForwardIterator \\_\\_last](#), [\\_OutputIterator \\_\\_↵out](#), [const \\_Distance \\_\\_n](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Member [\\_\\_gnu\\_cxx::random\\_sample\\_n](#) ([\\_ForwardIterator \\_\\_first](#), [\\_ForwardIterator \\_\\_last](#), [\\_OutputIterator \\_\\_↵out](#), [const \\_Distance \\_\\_n](#), [\\_RandomNumberGenerator &\\_\\_rand](#))

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class [\\_\\_gnu\\_cxx::rb\\_tree](#)<[\\_Key](#), [\\_Value](#), [\\_KeyOfValue](#), [\\_Compare](#), [\\_Alloc](#)>

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class [\\_\\_gnu\\_cxx::rope](#)<[\\_CharT](#), [\\_Alloc](#)>

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Class `__gnu_cxx::slist<_Tp, _Alloc>`

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

page **Mathematical Special Functions**

Provide accuracy comparisons on a per-function basis for a small number of targets.

Class `std::basic_string<_CharT, _Traits, _Alloc>`

\nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

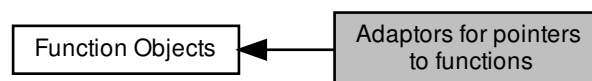
Member `std::regex_traits<_Ch_type>::transform_primary(_Fwd_iter __first, _Fwd_iter __last) const`

Implement this function correctly.

## 3 Module Documentation

### 3.1 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



#### Classes

- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`
- class `std::pointer_to_unary_function<_Arg, _Result>`

#### Functions

- template<typename \_Arg, typename \_Result>  
`pointer_to_unary_function<_Arg, _Result> std::ptr_fun(_Result(*__x)(_Arg))`
- template<typename \_Arg1, typename \_Arg2, typename \_Result>  
`pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun(_Result(*__x)(_Arg1, _Arg2))`

### 3.1.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

### 3.1.2 Function Documentation

#### 3.1.2.1 `ptr_fun()` [1/2]

```
template<typename _Arg , typename _Result >
pointer_to_unary_function<_Arg, _Result> std::ptr_fun (
    _Result(*) (_Arg) __x ) [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 1076 of file `stl_function.h`.

#### 3.1.2.2 `ptr_fun()` [2/2]

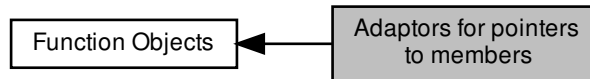
```
template<typename _Arg1 , typename _Arg2 , typename _Result >
pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (
    _Result(*) (_Arg1, _Arg2) __x ) [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 1102 of file `stl_function.h`.

## 3.2 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



### Classes

- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun_ref_t<_Ret, _Tp>`
- class `std::const_mem_fun_t<_Ret, _Tp>`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun_ref_t<_Ret, _Tp>`
- class `std::mem_fun_t<_Ret, _Tp>`

### Functions

- `template<typename _Ret, typename _Tp>`  
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`

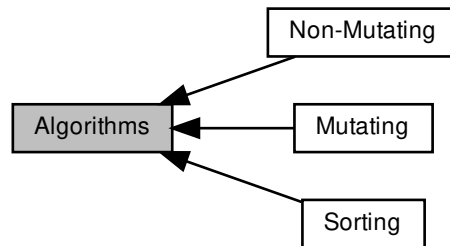
#### 3.2.1 Detailed Description

There are a total of  $8 = 2^3$  function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

### 3.3 Algorithms

Collaboration diagram for Algorithms:



#### Modules

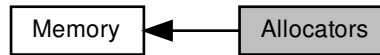
- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

#### 3.3.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

### 3.4 Allocators

Collaboration diagram for Allocators:



#### Classes

- struct `__gnu_cxx::__alloc_traits<_Alloc, typename >`
- class `__gnu_cxx::__mt_alloc<_Tp, _Poolp >`
- class `__gnu_cxx::__pool_alloc<_Tp >`
- class `__gnu_cxx::__ExtPtr_allocator<_Tp >`
- class `__gnu_cxx::array_allocator<_Tp, _Array >`
- class `__gnu_cxx::bitmap_allocator<_Tp >`
- class `__gnu_cxx::debug_allocator<_Alloc >`
- class `__gnu_cxx::malloc_allocator<_Tp >`
- class `__gnu_cxx::new_allocator<_Tp >`
- class `__gnu_cxx::throw_allocator_base<_Tp, _Cond >`
- class `std::allocator<_Tp >`
- class `std::allocator<void >`
- struct `std::allocator_traits<_Alloc >`
- class `std::scoped_allocator_adaptor<_OuterAlloc, _InnerAllocs >`
- struct `std::uses_allocator<_Tp, _Alloc >`

#### Typedefs

- template<typename `_Tp` >  
using `std::__allocator_base` = `__gnu_cxx::new_allocator<_Tp >`
- template<typename `_Alloc` >  
using `std::__outer_allocator_t` = `decltype(std::declval<_Alloc >().outer_allocator())`

#### Functions

- template<typename `_Alloc` >  
`__outermost_type<_Alloc >::type & std::__outermost` (`_Alloc &a`)
- template<typename `_T1`, typename `_T2` >  
`bool std::operator!=` (const `allocator<_T1 >` &, const `allocator<_T2 >` &) noexcept
- template<typename `_Tp` >  
`bool std::operator!=` (const `allocator<_Tp >` &, const `allocator<_Tp >` &) noexcept



- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool std::operator!= (const scoped\_allocator\_adaptor< _OutA1, _InA... > &__a, const scoped\_allocator\_adaptor< _OutA2, _InA... > &__b) noexcept`
- `template<typename _T1, typename _T2 >`  
`bool std::operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _Tp >`  
`bool std::operator== (const allocator< _Tp > &, const allocator< _Tp > &) noexcept`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool std::operator== (const scoped\_allocator\_adaptor< _OutA1, _InA... > &__a, const scoped\_allocator\_adaptor< _OutA2, _InA... > &__b) noexcept`

### 3.4.1 Detailed Description

Classes encapsulating memory operations.

### 3.4.2 Typedef Documentation

#### 3.4.2.1 `__allocator_base`

```
template<typename _Tp >  
using std::\_\_allocator\_base = typedef \_\_gnu\_cxx::new\_allocator<_Tp>
```

An alias to the base class for `std::allocator`.

Used to set the `std::allocator` base class to `__gnu_cxx::new_allocator`.

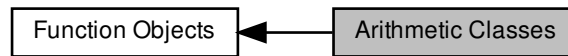
#### Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 48 of file `c++/allocator.h`.

### 3.5 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



#### Classes

- struct `std::divides< _Tp >`
- struct `std::divides< void >`
- struct `std::minus< _Tp >`
- struct `std::minus< void >`
- struct `std::modulus< _Tp >`
- struct `std::modulus< void >`
- struct `std::multiplies< _Tp >`
- struct `std::multiplies< void >`
- struct `std::negate< _Tp >`
- struct `std::negate< void >`
- struct `std::plus< _Tp >`

#### Macros

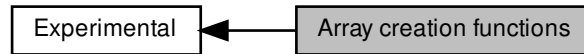
- `#define __cpp_lib_transparent_operators`

#### 3.5.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

### 3.6 Array creation functions

Collaboration diagram for Array creation functions:



#### Functions

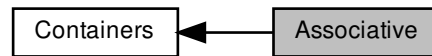
- `template<typename _Tp, size_t _Nm, size_t... _Idx>`  
`constexpr array< remove\_cv\_t< _Tp >, _Nm > std::experimental::fundamentals_v2::__to_array (_Tp(&__a)[_Nm], index\_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>`  
`constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::fundamentals_v2::make_array (_Types &&... __t)`
- `template<typename _Tp, size_t _Nm>`  
`constexpr array< remove\_cv\_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (_Tp(&__a)[_Nm]) noexcept(is\_nothrow\_constructible< remove\_cv\_t< _Tp >, _Tp & >::value)`

#### 3.6.1 Detailed Description

Array creation functions as described in N4529, Working Draft, C++ Extensions for Library Fundamentals, Version 2

### 3.7 Associative

Collaboration diagram for Associative:



#### Classes

- class `std::map<_Key, _Tp, _Compare, _Alloc >`
- class `std::multimap<_Key, _Tp, _Compare, _Alloc >`
- class `std::multiset<_Key, _Compare, _Alloc >`
- class `std::set<_Key, _Compare, _Alloc >`

#### 3.7.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

## 3.8 Atomics

### Classes

- struct `std::__atomic_base<_ITp>`
- struct `std::__atomic_base<_PTp*>`
- struct `std::__atomic_flag_base`
- struct `std::atomic<_Tp>`
- struct `std::atomic<_Tp*>`
- struct `std::atomic<bool>`
- struct `std::atomic<char>`
- struct `std::atomic<char16_t>`
- struct `std::atomic<char32_t>`
- struct `std::atomic<int>`
- struct `std::atomic<long>`
- struct `std::atomic<long long>`
- struct `std::atomic<short>`
- struct `std::atomic<signed char>`
- struct `std::atomic<unsigned char>`
- struct `std::atomic<unsigned int>`
- struct `std::atomic<unsigned long>`
- struct `std::atomic<unsigned long long>`
- struct `std::atomic<unsigned short>`
- struct `std::atomic<wchar_t>`
- struct `std::atomic_flag`

### Macros

- `#define ATOMIC_BOOL_LOCK_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_VAR_INIT(_VI)`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

## Typedefs

- typedef unsigned char **std::\_\_atomic\_flag\_data\_type**
- typedef [atomic](#)< bool > [std::atomic\\_bool](#)
- typedef [atomic](#)< char > [std::atomic\\_char](#)
- typedef [atomic](#)< char16\_t > [std::atomic\\_char16\\_t](#)
- typedef [atomic](#)< char32\_t > [std::atomic\\_char32\\_t](#)
- typedef [atomic](#)< int > [std::atomic\\_int](#)
- typedef [atomic](#)< int16\_t > [std::atomic\\_int16\\_t](#)
- typedef [atomic](#)< int32\_t > [std::atomic\\_int32\\_t](#)
- typedef [atomic](#)< int64\_t > [std::atomic\\_int64\\_t](#)
- typedef [atomic](#)< int8\_t > [std::atomic\\_int8\\_t](#)
- typedef [atomic](#)< int\_fast16\_t > [std::atomic\\_int\\_fast16\\_t](#)
- typedef [atomic](#)< int\_fast32\_t > [std::atomic\\_int\\_fast32\\_t](#)
- typedef [atomic](#)< int\_fast64\_t > [std::atomic\\_int\\_fast64\\_t](#)
- typedef [atomic](#)< int\_fast8\_t > [std::atomic\\_int\\_fast8\\_t](#)
- typedef [atomic](#)< int\_least16\_t > [std::atomic\\_int\\_least16\\_t](#)
- typedef [atomic](#)< int\_least32\_t > [std::atomic\\_int\\_least32\\_t](#)
- typedef [atomic](#)< int\_least64\_t > [std::atomic\\_int\\_least64\\_t](#)
- typedef [atomic](#)< int\_least8\_t > [std::atomic\\_int\\_least8\\_t](#)
- typedef [atomic](#)< intmax\_t > [std::atomic\\_intmax\\_t](#)
- typedef [atomic](#)< intptr\_t > [std::atomic\\_intptr\\_t](#)
- typedef [atomic](#)< long long > [std::atomic\\_llong](#)
- typedef [atomic](#)< long > [std::atomic\\_long](#)
- typedef [atomic](#)< ptrdiff\_t > [std::atomic\\_ptrdiff\\_t](#)
- typedef [atomic](#)< signed char > [std::atomic\\_schar](#)
- typedef [atomic](#)< short > [std::atomic\\_short](#)
- typedef [atomic](#)< size\_t > [std::atomic\\_size\\_t](#)
- typedef [atomic](#)< unsigned char > [std::atomic\\_uchar](#)
- typedef [atomic](#)< unsigned int > [std::atomic\\_uint](#)
- typedef [atomic](#)< uint16\_t > [std::atomic\\_uint16\\_t](#)
- typedef [atomic](#)< uint32\_t > [std::atomic\\_uint32\\_t](#)
- typedef [atomic](#)< uint64\_t > [std::atomic\\_uint64\\_t](#)
- typedef [atomic](#)< uint8\_t > [std::atomic\\_uint8\\_t](#)
- typedef [atomic](#)< uint\_fast16\_t > [std::atomic\\_uint\\_fast16\\_t](#)
- typedef [atomic](#)< uint\_fast32\_t > [std::atomic\\_uint\\_fast32\\_t](#)
- typedef [atomic](#)< uint\_fast64\_t > [std::atomic\\_uint\\_fast64\\_t](#)
- typedef [atomic](#)< uint\_fast8\_t > [std::atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic](#)< uint\_least16\_t > [std::atomic\\_uint\\_least16\\_t](#)
- typedef [atomic](#)< uint\_least32\_t > [std::atomic\\_uint\\_least32\\_t](#)
- typedef [atomic](#)< uint\_least64\_t > [std::atomic\\_uint\\_least64\\_t](#)
- typedef [atomic](#)< uint\_least8\_t > [std::atomic\\_uint\\_least8\\_t](#)
- typedef [atomic](#)< uintmax\_t > [std::atomic\\_uintmax\\_t](#)
- typedef [atomic](#)< uintptr\_t > [std::atomic\\_uintptr\\_t](#)
- typedef [atomic](#)< unsigned long long > [std::atomic\\_ullong](#)
- typedef [atomic](#)< unsigned long > [std::atomic\\_ulong](#)
- typedef [atomic](#)< unsigned short > [std::atomic\\_ushort](#)
- typedef [atomic](#)< wchar\_t > [std::atomic\\_wchar\\_t](#)
- typedef enum [std::memory\\_order](#) [std::memory\\_order](#)

## Enumerations

- enum `__memory_order_modifier` { `__memory_order_mask`, `__memory_order_modifier_mask`, `__memory_order_hle_acquire`, `__memory_order_hle_release` }
- enum `std::memory_order` { `memory_order_relaxed`, `memory_order_consume`, `memory_order_acquire`, `memory_order_release`, `memory_order_acq_rel`, `memory_order_seq_cst` }

## Functions

- `std::__attribute__((__always_inline__)) void atomic_thread_fence(memory_order __m) noexcept`
- `constexpr memory_order std::__cmpexch_failure_order(memory_order __m) noexcept`
- `constexpr memory_order std::__cmpexch_failure_order2(memory_order __m) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong(atomic<_ITp> *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong(volatile atomic<_ITp> *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit(atomic<_ITp> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit(volatile atomic<_ITp> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak(atomic<_ITp> *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak(volatile atomic<_ITp> *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit(atomic<_ITp> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit(volatile atomic<_ITp> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange(atomic<_ITp> *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange(volatile atomic<_ITp> *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit(atomic<_ITp> *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit(volatile atomic<_ITp> *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add(__atomic_base<_ITp> *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add(volatile __atomic_base<_ITp> *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add(volatile atomic<_ITp> * __a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add(atomic<_ITp> * __a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit(__atomic_base<_ITp> *__a, _ITp __i, memory_order __m) noexcept`

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m)`  
`noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noex-`  
`cept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m)`  
`noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`  
`noexcept`



- void **std::atomic\_flag\_clear** ([atomic\\_flag](#) \*\_\_a) noexcept
- void **std::atomic\_flag\_clear** (volatile [atomic\\_flag](#) \*\_\_a) noexcept
- void **std::atomic\_flag\_clear\_explicit** ([atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- void **std::atomic\_flag\_clear\_explicit** (volatile [atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- bool **std::atomic\_flag\_test\_and\_set** ([atomic\\_flag](#) \*\_\_a) noexcept
- bool **std::atomic\_flag\_test\_and\_set** (volatile [atomic\\_flag](#) \*\_\_a) noexcept
- bool **std::atomic\_flag\_test\_and\_set\_explicit** ([atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- bool **std::atomic\_flag\_test\_and\_set\_explicit** (volatile [atomic\\_flag](#) \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
void **std::atomic\_init** ([atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
void **std::atomic\_init** (volatile [atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
bool **std::atomic\_is\_lock\_free** (const [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
bool **std::atomic\_is\_lock\_free** (const volatile [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
\_ITp **std::atomic\_load** (const [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
\_ITp **std::atomic\_load** (const volatile [atomic](#)< \_ITp > \*\_\_a) noexcept
- template<typename \_ITp >  
\_ITp **std::atomic\_load\_explicit** (const [atomic](#)< \_ITp > \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
\_ITp **std::atomic\_load\_explicit** (const volatile [atomic](#)< \_ITp > \*\_\_a, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
void **std::atomic\_store** ([atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
void **std::atomic\_store** (volatile [atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i) noexcept
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** ([atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m) noexcept
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** (volatile [atomic](#)< \_ITp > \*\_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m) noexcept
- template<typename \_Tp >  
\_Tp **std::kill\_dependency** (\_Tp \_\_y) noexcept
- constexpr [memory\\_order](#) **std::operator&** ([memory\\_order](#) \_\_m, \_\_memory\_order\_modifier \_\_mod)
- constexpr [memory\\_order](#) **std::operator|** ([memory\\_order](#) \_\_m, \_\_memory\_order\_modifier \_\_mod)

### 3.8.1 Detailed Description

Components for performing atomic operations.

### 3.8.2 Macro Definition Documentation

### 3.8.2.1 ATOMIC\_BOOL\_LOCK\_FREE

```
#define ATOMIC_BOOL_LOCK_FREE
```

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

Definition at line 49 of file `atomic_lockfree_defines.h`.

## 3.8.3 Typedef Documentation

### 3.8.3.1 `atomic_bool`

```
typedef atomic<bool> std::atomic_bool
```

`atomic_bool`

Definition at line 868 of file `atomic`.

### 3.8.3.2 `atomic_char`

```
typedef atomic<char> std::atomic_char
```

`atomic_char`

Definition at line 871 of file `atomic`.

### 3.8.3.3 `atomic_char16_t`

```
typedef atomic<char16_t> std::atomic_char16_t
```

`atomic_char16_t`

Definition at line 907 of file `atomic`.

#### 3.8.3.4 atomic\_char32\_t

```
typedef atomic<char32_t> std::atomic_char32_t
```

atomic\_char32\_t

Definition at line 910 of file atomic.

#### 3.8.3.5 atomic\_int

```
typedef atomic<int> std::atomic_int
```

atomic\_int

Definition at line 886 of file atomic.

#### 3.8.3.6 atomic\_int16\_t

```
typedef atomic<int16_t> std::atomic_int16_t
```

atomic\_int16\_t

Definition at line 923 of file atomic.

#### 3.8.3.7 atomic\_int32\_t

```
typedef atomic<int32_t> std::atomic_int32_t
```

atomic\_int32\_t

Definition at line 929 of file atomic.

#### 3.8.3.8 atomic\_int64\_t

```
typedef atomic<int64_t> std::atomic_int64_t
```

atomic\_int64\_t

Definition at line 935 of file atomic.

### 3.8.3.9 atomic\_int8\_t

```
typedef atomic<int8_t> std::atomic_int8_t
```

atomic\_int8\_t

Definition at line 917 of file atomic.

### 3.8.3.10 atomic\_int\_fast16\_t

```
typedef atomic<int_fast16_t> std::atomic_int_fast16_t
```

atomic\_int\_fast16\_t

Definition at line 973 of file atomic.

### 3.8.3.11 atomic\_int\_fast32\_t

```
typedef atomic<int_fast32_t> std::atomic_int_fast32_t
```

atomic\_int\_fast32\_t

Definition at line 979 of file atomic.

### 3.8.3.12 atomic\_int\_fast64\_t

```
typedef atomic<int_fast64_t> std::atomic_int_fast64_t
```

atomic\_int\_fast64\_t

Definition at line 985 of file atomic.

### 3.8.3.13 atomic\_int\_fast8\_t

```
typedef atomic<int_fast8_t> std::atomic_int_fast8_t
```

atomic\_int\_fast8\_t

Definition at line 967 of file atomic.

#### 3.8.3.14 `atomic_int_least16_t`

```
typedef atomic<int_least16_t> std::atomic_int_least16_t
```

`atomic_int_least16_t`

Definition at line 948 of file `atomic`.

#### 3.8.3.15 `atomic_int_least32_t`

```
typedef atomic<int_least32_t> std::atomic_int_least32_t
```

`atomic_int_least32_t`

Definition at line 954 of file `atomic`.

#### 3.8.3.16 `atomic_int_least64_t`

```
typedef atomic<int_least64_t> std::atomic_int_least64_t
```

`atomic_int_least64_t`

Definition at line 960 of file `atomic`.

#### 3.8.3.17 `atomic_int_least8_t`

```
typedef atomic<int_least8_t> std::atomic_int_least8_t
```

`atomic_int_least8_t`

Definition at line 942 of file `atomic`.

#### 3.8.3.18 `atomic_intmax_t`

```
typedef atomic<intmax_t> std::atomic_intmax_t
```

`atomic_intmax_t`

Definition at line 1006 of file `atomic`.

#### 3.8.3.19 atomic\_intptr\_t

```
typedef atomic<intptr_t> std::atomic_intptr_t
```

atomic\_intptr\_t

Definition at line 993 of file atomic.

#### 3.8.3.20 atomic\_llong

```
typedef atomic<long long> std::atomic_llong
```

atomic\_llong

Definition at line 898 of file atomic.

#### 3.8.3.21 atomic\_long

```
typedef atomic<long> std::atomic_long
```

atomic\_long

Definition at line 892 of file atomic.

#### 3.8.3.22 atomic\_ptrdiff\_t

```
typedef atomic<ptrdiff_t> std::atomic_ptrdiff_t
```

atomic\_ptrdiff\_t

Definition at line 1002 of file atomic.

#### 3.8.3.23 atomic\_schar

```
typedef atomic<signed char> std::atomic_schar
```

atomic\_schar

Definition at line 874 of file atomic.

#### 3.8.3.24 atomic\_short

```
typedef atomic<short> std::atomic_short
```

atomic\_short

Definition at line 880 of file atomic.

#### 3.8.3.25 atomic\_size\_t

```
typedef atomic<size_t> std::atomic_size_t
```

atomic\_size\_t

Definition at line 999 of file atomic.

#### 3.8.3.26 atomic\_uchar

```
typedef atomic<unsigned char> std::atomic_uchar
```

atomic\_uchar

Definition at line 877 of file atomic.

#### 3.8.3.27 atomic\_uint

```
typedef atomic<unsigned int> std::atomic_uint
```

atomic\_uint

Definition at line 889 of file atomic.

#### 3.8.3.28 atomic\_uint16\_t

```
typedef atomic<uint16_t> std::atomic_uint16_t
```

atomic\_uint16\_t

Definition at line 926 of file atomic.

### 3.8.3.29 atomic\_uint32\_t

```
typedef atomic<uint32_t> std::atomic_uint32_t
```

atomic\_uint32\_t

Definition at line 932 of file atomic.

### 3.8.3.30 atomic\_uint64\_t

```
typedef atomic<uint64_t> std::atomic_uint64_t
```

atomic\_uint64\_t

Definition at line 938 of file atomic.

### 3.8.3.31 atomic\_uint8\_t

```
typedef atomic<uint8_t> std::atomic_uint8_t
```

atomic\_uint8\_t

Definition at line 920 of file atomic.

### 3.8.3.32 atomic\_uint\_fast16\_t

```
typedef atomic<uint_fast16_t> std::atomic_uint_fast16_t
```

atomic\_uint\_fast16\_t

Definition at line 976 of file atomic.

### 3.8.3.33 atomic\_uint\_fast32\_t

```
typedef atomic<uint_fast32_t> std::atomic_uint_fast32_t
```

atomic\_uint\_fast32\_t

Definition at line 982 of file atomic.



**3.8.3.34 atomic\_uint\_fast64\_t**

```
typedef atomic<uint_fast64_t> std::atomic_uint_fast64_t
```

atomic\_uint\_fast64\_t

Definition at line 988 of file atomic.

**3.8.3.35 atomic\_uint\_fast8\_t**

```
typedef atomic<uint_fast8_t> std::atomic_uint_fast8_t
```

atomic\_uint\_fast8\_t

Definition at line 970 of file atomic.

**3.8.3.36 atomic\_uint\_least16\_t**

```
typedef atomic<uint_least16_t> std::atomic_uint_least16_t
```

atomic\_uint\_least16\_t

Definition at line 951 of file atomic.

**3.8.3.37 atomic\_uint\_least32\_t**

```
typedef atomic<uint_least32_t> std::atomic_uint_least32_t
```

atomic\_uint\_least32\_t

Definition at line 957 of file atomic.

**3.8.3.38 atomic\_uint\_least64\_t**

```
typedef atomic<uint_least64_t> std::atomic_uint_least64_t
```

atomic\_uint\_least64\_t

Definition at line 963 of file atomic.

#### 3.8.3.39 atomic\_uint\_least8\_t

```
typedef atomic<uint_least8_t> std::atomic_uint_least8_t
```

atomic\_uint\_least8\_t

Definition at line 945 of file atomic.

#### 3.8.3.40 atomic\_uintmax\_t

```
typedef atomic<uintmax_t> std::atomic_uintmax_t
```

atomic\_uintmax\_t

Definition at line 1009 of file atomic.

#### 3.8.3.41 atomic\_uintptr\_t

```
typedef atomic<uintptr_t> std::atomic_uintptr_t
```

atomic\_uintptr\_t

Definition at line 996 of file atomic.

#### 3.8.3.42 atomic\_ullong

```
typedef atomic<unsigned long long> std::atomic_ullong
```

atomic\_ullong

Definition at line 901 of file atomic.

#### 3.8.3.43 atomic\_ulong

```
typedef atomic<unsigned long> std::atomic_ulong
```

atomic\_ulong

Definition at line 895 of file atomic.

#### 3.8.3.44 atomic\_ushort

```
typedef atomic<unsigned short> std::atomic_ushort
```

atomic\_ushort

Definition at line 883 of file atomic.

#### 3.8.3.45 atomic\_wchar\_t

```
typedef atomic<wchar_t> std::atomic_wchar_t
```

atomic\_wchar\_t

Definition at line 904 of file atomic.

#### 3.8.3.46 memory\_order

```
typedef enum std::memory_order std::memory_order
```

Enumeration for memory\_order.

### 3.8.4 Enumeration Type Documentation

#### 3.8.4.1 memory\_order

```
enum std::memory_order
```

Enumeration for memory\_order.

Definition at line 55 of file atomic\_base.h.

### 3.8.5 Function Documentation

#### 3.8.5.1 kill\_dependency()

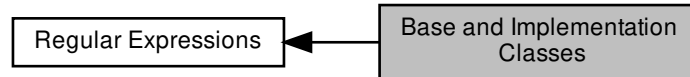
```
template<typename _Tp >  
_Tp std::kill_dependency (   
    _Tp __y ) [inline], [noexcept]
```

kill\_dependency

Definition at line 111 of file atomic\_base.h.

### 3.9 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



#### Classes

- struct `std::__detail::BracketMatcher<_TraitsT, __icase, __collate >`
- class `std::__detail::Compiler<_TraitsT >`
- class `std::__detail::Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`
- class `std::__detail::Scanner<_CharT >`
- class `std::__detail::StateSeq<_TraitsT >`

#### Typedefs

- template<typename \_Iter, typename \_TraitsT >  
using `std::__detail::__disable_if_contiguous_normal_iter` = typename `enable_if< !__is_contiguous_↵  
normal_iter<_Iter >::value, std::shared_ptr< const _NFA<_TraitsT > > >::type`
- template<typename \_Iter, typename \_TraitsT >  
using `std::__detail::__enable_if_contiguous_normal_iter` = typename `enable_if< __is_contiguous_normal_↵  
_iter<_Iter >::value, std::shared_ptr< const _NFA<_TraitsT > > >::type`
- template<typename \_CharT >  
using `std::__detail::Matcher` = `std::function< bool(_CharT)>`
- typedef long `std::__detail::StateIdT`

#### Enumerations

- enum `std::__detail::Opcode` : int {  
    `_S_opcode_unknown`, `_S_opcode_alternative`, `_S_opcode_repeat`, `_S_opcode_backref`,  
    `_S_opcode_line_begin_assertion`, `_S_opcode_line_end_assertion`, `_S_opcode_word_boundary`, `_S_↵  
opcode_subexpr_lookahead`,  
    `_S_opcode_subexpr_begin`, `_S_opcode_subexpr_end`, `_S_opcode_dummy`, `_S_opcode_match`,  
    `_S_opcode_accept` }

#### Functions

- template<typename \_TraitsT, typename \_FwdIter >  
    `enable_if_contiguous_normal_iter<_FwdIter, _TraitsT > std::__detail::__compile_nfa` (`_FwdIter __first`, `_↵  
FwdIter __last`, `const typename _TraitsT::locale_type &__loc`, `regex_constants::syntax_option_type __flags`)

## Variables

- static const \_StateIdT **std::\_\_detail::\_S\_invalid\_state\_id**

### 3.9.1 Detailed Description

### 3.9.2 Enumeration Type Documentation

#### 3.9.2.1 \_Opcode

```
enum std::__detail::_Opcode : int
```

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

Definition at line 56 of file regex\_automaton.h.

### 3.10 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



#### Classes

- struct `std::__detail::Default_ranged_hash`
- struct `std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >`
- struct `std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >`
- struct `std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >`
- struct `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- struct `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- struct `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::Equality_base`
- struct `std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- struct `std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, Default_ranged_hash, false >`
- struct `std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, Default_ranged_hash, true >`
- struct `std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`
- struct `std::__detail::Hash_node< _Value, _Cache_hash_code >`
- struct `std::__detail::Hash_node< _Value, false >`
- struct `std::__detail::Hash_node< _Value, true >`
- struct `std::__detail::Hash_node_base`
- struct `std::__detail::Hash_node_value_base< _Value >`
- struct `std::__detail::Hashtable_alloc< _NodeAlloc >`
- struct `std::__detail::Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`
- struct `std::__detail::Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >`
- struct `std::__detail::Hashtable_ebo_helper< _Nm, _Tp, false >`
- struct `std::__detail::Hashtable_ebo_helper< _Nm, _Tp, true >`
- struct `std::__detail::Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >`
- struct `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators >`
- struct `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- struct `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- struct `std::__detail::Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- struct `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`
- struct `std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- struct `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`

- struct `std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::_Mask_range_hashing`
- struct `std::__detail::_Mod_range_hashing`
- struct `std::__detail::_Node_const_iterator<_Value, __constant_iterators, __cache >`
- struct `std::__detail::_Node_iterator<_Value, __constant_iterators, __cache >`
- struct `std::__detail::_Node_iterator_base<_Value, _Cache_hash_code >`
- struct `std::__detail::_Power2_rehash_policy`
- struct `std::__detail::_Prime_rehash_policy`
- struct `std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename... >`
- struct `std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::false_type >`
- struct `std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::true_type >`
- class `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

## Typedefs

- template<typename \_Policy >  
using `std::__detail::__has_load_factor` = typename \_Policy::\_\_has\_load\_factor
- template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_H1, typename \_H2, typename \_Hash >  
using `std::__detail::__hash_code_for_local_iter` = \_Hash\_code\_storage<\_Hash\_code\_base<\_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false > >

## Functions

- `_GLIBCXX14_CONSTEXPR std::size_t std::__detail::__clp2 (std::size_t __n) noexcept`
- template<class \_Iterator >  
`std::iterator_traits<_Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- template<class \_Iterator >  
`std::iterator_traits<_Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- template<class \_Iterator >  
`std::iterator_traits<_Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last)`
- `__bucket_type * std::__detail::_Hashtable_alloc<_NodeAlloc >::_M_allocate_buckets (std::size_t __n)`
- template<typename... \_Args>  
`__node_type * std::__detail::_Hashtable_alloc<_NodeAlloc >::_M_allocate_node (_Args &&... __args)`
- void `std::__detail::_Hashtable_alloc<_NodeAlloc >::_M_deallocate_buckets (__bucket_type *, std::size_t __n)`
- void `std::__detail::_Hashtable_alloc<_NodeAlloc >::_M_deallocate_node (__node_type * __n)`
- void `std::__detail::_Hashtable_alloc<_NodeAlloc >::_M_deallocate_nodes (__node_type * __n)`
- bool `std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::_M_equal (const __hashtable &) const`
- bool `std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >::_M_equal (const __hashtable &) const`
- template<typename \_InputIterator, typename \_NodeGetter >  
void `std::__detail::_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::_M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &, true_type)`
- template<typename \_InputIterator, typename \_NodeGetter >  
void `std::__detail::_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::_M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &, false_type)`

- `template<typename _Uiterator >`  
`static bool std::__detail::Equality_base::_S_is_permutation ( _Uiterator, _Uiterator, _Uiterator)`
- `mapped_type & std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _↵`  
`RehashPolicy, _Traits, true >::at (const key_type &__k)`
- `const mapped_type & std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash,`  
`_RehashPolicy, _Traits, true >::at (const key_type &__k) const`
- `template<typename _Value , bool _Cache_hash_code>`  
`bool std::__detail::operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const`  
`_Node_iterator_base< _Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>`  
`bool std::__detail::operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, _↵`  
`_cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _Value , bool _Cache_hash_code>`  
`bool std::__detail::operator== (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const`  
`_Node_iterator_base< _Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>`  
`bool std::__detail::operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, _↵`  
`_cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `mapped_type & std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _↵`  
`RehashPolicy, _Traits, true >::operator[] (const key_type &__k)`
- `mapped_type & std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _↵`  
`RehashPolicy, _Traits, true >::operator[] (key_type &&__k)`

### 3.10.1 Detailed Description

### 3.10.2 Function Documentation

#### 3.10.2.1 \_\_clp2()

```
_GLIBCXX14_CONSTEXPR std::size_t std::__detail::__clp2 (
    std::size_t __n ) [inline], [noexcept]
```

Compute closest power of 2.

Definition at line 510 of file hashtable\_policy.h.



### 3.11 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



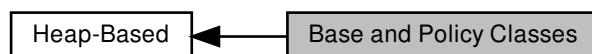
#### Classes

- [class `\_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >`](#)
- [class `\_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >`](#)
- [class `\_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >`](#)
- [class `\_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >`](#)

#### 3.11.1 Detailed Description

### 3.12 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



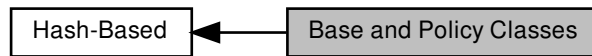
#### Classes

- `class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`
- `class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`
- `class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`
- `class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`
- `class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`

#### 3.12.1 Detailed Description

### 3.13 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



#### Classes

- [class `\_\_gnu\_pbds::detail::cc\_ht\_map`](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >
- [class `\_\_gnu\_pbds::detail::gp\_ht\_map`](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >

#### 3.13.1 Detailed Description

## 3.14 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



### Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution< _IntType >`
- class `std::geometric_distribution< _IntType >`
- class `std::negative_binomial_distribution< _IntType >`

### Functions

- bool `std::operator!=` (const `std::bernoulli_distribution` &\_\_d1, const `std::bernoulli_distribution` &\_\_d2)
- template<typename `_IntType` >  
bool `std::operator!=` (const `std::binomial_distribution< _IntType >` &\_\_d1, const `std::binomial_distribution< _IntType >` &\_\_d2)
- template<typename `_IntType` >  
bool `std::operator!=` (const `std::geometric_distribution< _IntType >` &\_\_d1, const `std::geometric_distribution< _IntType >` &\_\_d2)
- template<typename `_IntType` >  
bool `std::operator!=` (const `std::negative_binomial_distribution< _IntType >` &\_\_d1, const `std::negative_binomial_distribution< _IntType >` &\_\_d2)
- template<typename `_CharT`, typename `_Traits` >  
`std::basic_ostream< _CharT, _Traits >` & `std::operator<<` (`std::basic_ostream< _CharT, _Traits >` &\_\_os, const `std::bernoulli_distribution` &\_\_x)
- template<typename `_IntType`, typename `_CharT`, typename `_Traits` >  
`std::basic_ostream< _CharT, _Traits >` & `std::operator<<` (`std::basic_ostream< _CharT, _Traits >` &\_\_os, const `std::geometric_distribution< _IntType >` &\_\_x)
- template<typename `_CharT`, typename `_Traits` >  
`std::basic_istream< _CharT, _Traits >` & `std::operator>>` (`std::basic_istream< _CharT, _Traits >` &\_\_is, `std::bernoulli_distribution` &\_\_x)
- template<typename `_IntType`, typename `_CharT`, typename `_Traits` >  
`std::basic_istream< _CharT, _Traits >` & `std::operator>>` (`std::basic_istream< _CharT, _Traits >` &\_\_is, `std::geometric_distribution< _IntType >` &\_\_x)

#### 3.14.1 Detailed Description

#### 3.14.2 Function Documentation

#### 3.14.2.1 `operator!=( )` [1/4]

```
bool std::operator!=(  
    const std::bernoulli_distribution & __d1,  
    const std::bernoulli_distribution & __d2 ) [inline]
```

Return true if two Bernoulli distributions have different parameters.

Definition at line 3614 of file random.h.

#### 3.14.2.2 `operator!=( )` [2/4]

```
template<typename _IntType >  
bool std::operator!=(  
    const std::binomial_distribution< _IntType > & __d1,  
    const std::binomial_distribution< _IntType > & __d2 ) [inline]
```

Return true if two binomial distributions are different.

Definition at line 3885 of file random.h.

#### 3.14.2.3 `operator!=( )` [3/4]

```
template<typename _IntType >  
bool std::operator!=(  
    const std::geometric_distribution< _IntType > & __d1,  
    const std::geometric_distribution< _IntType > & __d2 ) [inline]
```

Return true if two geometric distributions have different parameters.

Definition at line 4059 of file random.h.

#### 3.14.2.4 `operator!=( )` [4/4]

```
template<typename _IntType >  
bool std::operator!=(  
    const std::negative_binomial_distribution< _IntType > & __d1,  
    const std::negative_binomial_distribution< _IntType > & __d2 ) [inline]
```

Return true if two negative binomial distributions are different.

Definition at line 4309 of file random.h.

#### 3.14.2.5 `operator<<( )` [1/2]

```
template<typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (  
    std::basic_ostream< _CharT, _Traits > & __os,  
    const std::bernoulli_distribution & __x )
```

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 995 of file `bits/random.tcc`.

References `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::bernoulli_distribution::p()`, `std::ios_base::precision()`, `std::scientific()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

3.14.2.6 `operator<<()` [2/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::geometric_distribution< _IntType > & __x )
```

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>geometric_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1077 of file `bits/random.tcc`.

References `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::geometric_distribution< _IntType >::p()`, `std::ios_base::precision()`, `std::scientific()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

3.14.2.7 `operator>>()` [1/2]

```
template<typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::bernoulli_distribution & __x )
```

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

$\leftrightarrow$ __is	An input stream.
$\leftrightarrow$ __x	A bernoulli_distribution random number generator engine.

**Returns**

The input stream with \_\_x extracted or in an error state.

Definition at line 3644 of file random.h.

References std::bernoulli\_distribution::param().

**3.14.2.8 operator>>() [2/2]**

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::geometric_distribution< _IntType > & __x )
```

Extracts a geometric\_distribution random number distribution \_\_x from the input stream \_\_is.

**Parameters**

$\leftrightarrow$ __is	An input stream.
$\leftrightarrow$ __x	A geometric_distribution random number generator engine.

**Returns**

The input stream with \_\_x extracted or in an error state.

Definition at line 1101 of file bits/random.tcc.

References std::ios\_base::flags(), std::geometric\_distribution< \_IntType >::param(), and std::skipws().

## 3.15 Binary Search

Collaboration diagram for Binary Search:



### Functions

- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

### 3.15.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal\_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.



### 3.15.2 Function Documentation

#### 3.15.2.1 `binary_search()` [1/2]

```
template<typename _ForwardIterator , typename _Tp >
bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val )
```

Determines whether an element exists in a range.

##### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

##### Returns

True if `__val` (or its equivalent) is in `[__first,__last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2247 of file `stl_algo.h`.

#### 3.15.2.2 `binary_search()` [2/2]

```
template<typename _ForwardIterator , typename _Tp , typename _Compare >
bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp )
```

Determines whether an element exists in a range.

##### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

True if `__val` (or its equivalent) is in `[__first,__last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2280 of file `stl_algo.h`.

**3.15.2.3 `equal_range()`** [1/2]

```
template<typename _ForwardIterator , typename _Tp >
pair<_ForwardIterator, _ForwardIterator> std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val ) [inline]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

Definition at line 2178 of file `stl_algo.h`.

**3.15.2.4 `equal_range()`** [2/2]

```
template<typename _ForwardIterator , typename _Tp , typename _Compare >
pair<_ForwardIterator, _ForwardIterator> std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp ) [inline]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),  
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

Definition at line 2214 of file `stl_algo.h`.

**3.15.2.5 lower\_bound() [1/2]**

```
template<typename _ForwardIterator , typename _Tp >  
_ForwardIterator std::lower_bound (  
    _ForwardIterator __first,  
    _ForwardIterator __last,  
    const _Tp & __val ) [inline]
```

Finds the first position in which *val* could be inserted without changing the ordering.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

**Returns**

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

Definition at line 984 of file `stl_algobase.h`.

**3.15.2.6 lower\_bound() [2/2]**

```
template<typename _ForwardIterator , typename _Tp , typename _Compare >  
_ForwardIterator std::lower_bound (  

```

```

_FowardIterator __first,
_FowardIterator __last,
const _Tp & __val,
_Compare __comp ) [inline]

```

Finds the first position in which `__val` could be inserted without changing the ordering.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

#### Returns

An iterator pointing to the first element *not less than* `__val`, or `end()` if every element is less than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2023 of file `stl_algo.h`.

#### 3.15.2.7 `upper_bound()` [1/2]

```

template<typename _ForwardIterator , typename _Tp >
_FowardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val ) [inline]

```

Finds the last position in which `__val` could be inserted without changing the ordering.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

#### Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

Definition at line 2077 of file `stl_algo.h`.

### 3.15.2.8 upper\_bound() [2/2]

```
template<typename _ForwardIterator , typename _Tp , typename _Compare >
_FowardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp ) [inline]
```

Finds the last position in which `__val` could be inserted without changing the ordering.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

#### Returns

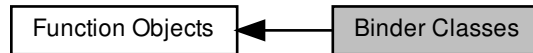
An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2107 of file `stl_algo.h`.

## 3.16 Binder Classes

Collaboration diagram for Binder Classes:



### Namespaces

- [std::placeholders](#)

### Classes

- struct [std::\\_Placeholder<\\_Num>](#)
- class [std::binder1st<\\_Operation>](#)
- class [std::binder2nd<\\_Operation>](#)
- struct [std::is\\_bind\\_expression<\\_Tp>](#)
- struct [std::is\\_bind\\_expression<\\_Bind<\\_Signature>>](#)
- struct [std::is\\_bind\\_expression<\\_Bind\\_result<\\_Result, \\_Signature>>](#)
- struct [std::is\\_bind\\_expression<const \\_Bind<\\_Signature>>](#)
- struct [std::is\\_bind\\_expression<const \\_Bind\\_result<\\_Result, \\_Signature>>](#)
- struct [std::is\\_bind\\_expression<const volatile \\_Bind<\\_Signature>>](#)
- struct [std::is\\_bind\\_expression<const volatile \\_Bind\\_result<\\_Result, \\_Signature>>](#)
- struct [std::is\\_bind\\_expression<volatile \\_Bind<\\_Signature>>](#)
- struct [std::is\\_bind\\_expression<volatile \\_Bind\\_result<\\_Result, \\_Signature>>](#)
- struct [std::is\\_placeholder<\\_Tp>](#)
- struct [std::is\\_placeholder<\\_Placeholder<\\_Num>>](#)

### Functions

- template<typename \_Func, typename... \_BoundArgs>  
[\\_Bind\\_helper<\\_\\_is\\_socketlike<\\_Func>::value, \\_Func, \\_BoundArgs...>::type](#) [std::bind](#) (\_Func &&\_\_f, \_BoundArgs &&... \_\_args)
- template<typename \_Result, typename \_Func, typename... \_BoundArgs>  
[\\_Bindres\\_helper<\\_Result, \\_Func, \\_BoundArgs...>::type](#) [std::bind](#) (\_Func &&\_\_f, \_BoundArgs &&... \_\_args)
- template<typename \_Operation, typename \_Tp>  
[binder1st<\\_Operation>](#) [std::bind1st](#) (const \_Operation &\_\_fn, const \_Tp &\_\_x)
- template<typename \_Operation, typename \_Tp>  
[binder2nd<\\_Operation>](#) [std::bind2nd](#) (const \_Operation &\_\_fn, const \_Tp &\_\_x)

### 3.16.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B's operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>(), 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(), 5)`.

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::placeholders::_1, x)`.

### 3.16.2 Function Documentation

#### 3.16.2.1 `bind()` [1/2]

```
template<typename _Func , typename... _BoundArgs>
_Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs...>::type std::bind (
    _Func && __f,
    _BoundArgs &&... __args ) [inline]
```

Function template for `std::bind`.

Definition at line 808 of file `functional`.

#### 3.16.2.2 `bind()` [2/2]

```
template<typename _Result , typename _Func , typename... _BoundArgs>
_Bindres_helper<_Result, _Func, _BoundArgs...>::type std::bind (
    _Func && __f,
    _BoundArgs &&... __args ) [inline]
```

Function template for `std::bind<R>`.

Definition at line 832 of file `functional`.

### 3.16.2.3 bind1st()

```
template<typename _Operation , typename _Tp >
binder1st<_Operation> std::bind1st (
    const _Operation & __fn,
    const _Tp & __x ) [inline]
```

One of the [binder functors](#).

Definition at line 135 of file binders.h.

### 3.16.2.4 bind2nd()

```
template<typename _Operation , typename _Tp >
binder2nd<_Operation> std::bind2nd (
    const _Operation & __fn,
    const _Tp & __x ) [inline]
```

One of the [binder functors](#).

Definition at line 170 of file binders.h.



### 3.17 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



#### Classes

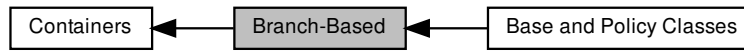
- struct `std::logical_and< _Tp >`
- struct `std::logical_and< void >`
- struct `std::logical_not< _Tp >`
- struct `std::logical_not< void >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< void >`

#### 3.17.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

### 3.18 Branch-Based

Collaboration diagram for Branch-Based:



#### Modules

- [Base and Policy Classes](#)

#### Classes

- [class `\_\_gnu\_pbds::basic\_branch< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >`](#)
- [class `\_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >`](#)
- [class `\_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >`](#)

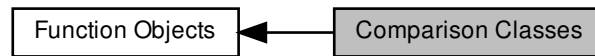
#### Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

#### 3.18.1 Detailed Description

### 3.19 Comparison Classes

Collaboration diagram for Comparison Classes:



#### Classes

- struct `std::equal_to<_Tp>`
- struct `std::equal_to<void>`
- struct `std::greater<_Tp>`
- struct `std::greater<void>`
- struct `std::greater_equal<_Tp>`
- struct `std::greater_equal<void>`
- struct `std::less<_Tp>`
- struct `std::less<void>`
- struct `std::less_equal<_Tp>`
- struct `std::less_equal<void>`
- struct `std::not_equal_to<_Tp>`
- struct `std::not_equal_to<void>`

#### 3.19.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

## 3.20 Complex Numbers

Collaboration diagram for Complex Numbers:



### Classes

- struct `std::complex< _Tp >`
- struct `std::complex< double >`
- struct `std::complex< float >`
- struct `std::complex< long double >`

### Functions

- constexpr `std::complex< float >::complex` (const `complex< double >` &)
- constexpr `std::complex< float >::complex` (const `complex< long double >` &)
- constexpr `std::complex< double >::complex` (const `complex< long double >` &)
- template<typename `_Tp` >  
`_Tp std::__complex_abs` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`_Tp std::__complex_arg` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_cos` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_cosh` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_exp` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_log` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_pow` (const `complex< _Tp >` &\_\_x, const `complex< _Tp >` &\_\_y)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_pow_unsigned` (`complex< _Tp >` \_\_x, unsigned \_\_n)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_sin` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_sinh` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_sqrt` (const `complex< _Tp >` &\_\_z)
- template<typename `_Tp` >  
`complex< _Tp > std::__complex_tan` (const `complex< _Tp >` &\_\_z)

- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &)`
- `complex< _Tp > & std::complex< _Tp >::operator/= (const _Tp &)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex< _Tp >::operator/= (const complex< _Up > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `complex< _Tp > & std::complex< _Tp >::operator= (const _Tp &)`
- `template<typename _Up >`  
`complex< _Tp > & std::complex< _Tp >::operator= (const complex< _Up > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`

- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar` (const \_Tp &\_\_rho,  
const \_Up &\_\_theta)
- `template<typename _Tp >`  
`complex< _Tp > std::pow` (const `complex< _Tp >` &, int)
- `template<typename _Tp >`  
`complex< _Tp > std::pow` (const `complex< _Tp >` &, const \_Tp &)
- `template<typename _Tp >`  
`complex< _Tp > std::pow` (const `complex< _Tp >` &, const `complex< _Tp >` &)
- `template<typename _Tp >`  
`complex< _Tp > std::pow` (const \_Tp &, const `complex< _Tp >` &)
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow` (const `std::complex<`  
`_Tp >` &\_\_x, const \_Up &\_\_y)
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow` (const \_Tp &\_\_x,  
const `std::complex< _Tp >` &\_\_y)
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow` (const `std::complex<`  
`_Tp >` &\_\_x, const `std::complex< _Tp >` &\_\_y)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow` (const `std::complex< _Tp >` &\_\_x, const \_Tp &\_\_y)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow` (const \_Tp &\_\_x, const `std::complex< _Tp >` &\_\_y)
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow` (const `std::complex< _Tp >` &\_\_x, const `std::complex< _Tp >` &\_\_y)
- `template<typename _Tp >`  
`constexpr _Tp std::real` (const `complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`complex< _Tp > std::sin` (const `complex< _Tp >` &)
- `template<typename _Tp >`  
`complex< _Tp > std::sinh` (const `complex< _Tp >` &)
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt` (const `complex< _Tp >` &)
- `template<typename _Tp >`  
`complex< _Tp > std::tan` (const `complex< _Tp >` &)
- `template<typename _Tp >`  
`complex< _Tp > std::tanh` (const `complex< _Tp >` &)
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator+` (const `complex< _Tp >` &\_\_x, const `complex< _Tp >` &\_\_y)
- `template<typename _Tp >`  
`complex< _Tp > std::operator+` (const `complex< _Tp >` &\_\_x, const \_Tp &\_\_y)
- `template<typename _Tp >`  
`complex< _Tp > std::operator+` (const \_Tp &\_\_x, const `complex< _Tp >` &\_\_y)
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator-` (const `complex< _Tp >` &\_\_x, const `complex< _Tp >` &\_\_y)
- `template<typename _Tp >`  
`complex< _Tp > std::operator-` (const `complex< _Tp >` &\_\_x, const \_Tp &\_\_y)
- `template<typename _Tp >`  
`complex< _Tp > std::operator-` (const \_Tp &\_\_x, const `complex< _Tp >` &\_\_y)

- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

### 3.20.1 Detailed Description

Classes and functions for complex numbers.

### 3.20.2 Function Documentation

#### 3.20.2.1 `abs()`

```
template<typename _Tp >
_Tp std::abs (
    const complex< _Tp > & __z ) [inline]
```

Return magnitude of `z`.

Definition at line 622 of file `complex`.

Referenced by `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

### 3.20.2.2 `arg()`

```
template<typename _Tp >
_Tp std::arg (
    const complex< _Tp > & __z ) [inline]
```

Return phase angle of  $z$ .

Definition at line 649 of file `complex`.

### 3.20.2.3 `conj()`

```
template<typename _Tp >
complex< _Tp > std::conj (
    const complex< _Tp > & __z ) [inline]
```

Return complex conjugate of  $z$ .

Definition at line 698 of file `complex`.

### 3.20.2.4 `cos()`

```
template<typename _Tp >
complex< _Tp > std::cos (
    const complex< _Tp > & __z ) [inline]
```

Return complex cosine of  $z$ .

Definition at line 730 of file `complex`.

### 3.20.2.5 `cosh()`

```
template<typename _Tp >
complex< _Tp > std::cosh (
    const complex< _Tp > & __z ) [inline]
```

Return complex hyperbolic cosine of  $z$ .

Definition at line 760 of file `complex`.



### 3.20.2.6 exp()

```
template<typename _Tp >
complex< _Tp > std::exp (
    const complex< _Tp > & __z ) [inline]
```

Return complex base e exponential of z.

Definition at line 786 of file complex.

### 3.20.2.7 fabs()

```
template<typename _Tp >
std::complex< _Tp > std::tr1::fabs (
    const std::complex< _Tp > & __z ) [inline]
```

fabs(\_\_z) [8.1.8].

Definition at line 309 of file tr1/complex.

### 3.20.2.8 log()

```
template<typename _Tp >
complex< _Tp > std::log (
    const complex< _Tp > & __z ) [inline]
```

Return complex natural logarithm of z.

Definition at line 813 of file complex.

Referenced by `std::generate_canonical()`, `std::normal_distribution< result_type >::operator()()`, `std::gamma_distribution< result_type >::operator()()`, `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

### 3.20.2.9 log10()

```
template<typename _Tp >
complex< _Tp > std::log10 (
    const complex< _Tp > & __z ) [inline]
```

Return complex base 10 logarithm of z.

Definition at line 818 of file complex.

#### 3.20.2.10 norm()

```
template<typename _Tp >
_Tp std::norm (
    const complex< _Tp > & __z ) [inline]
```

Return  $z$  magnitude squared.

Definition at line 682 of file complex.

#### 3.20.2.11 operator!=( ) [1/3]

```
template<typename _Tp >
constexpr bool std::operator!= (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return false if  $x$  is equal to  $y$ .

Definition at line 476 of file complex.

#### 3.20.2.12 operator!=( ) [2/3]

```
template<typename _Tp >
constexpr bool std::operator!= (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return false if  $x$  is equal to  $y$ .

Definition at line 481 of file complex.

#### 3.20.2.13 operator!=( ) [3/3]

```
template<typename _Tp >
constexpr bool std::operator!= (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return false if  $x$  is equal to  $y$ .

Definition at line 486 of file complex.

**3.20.2.14 operator\*()** [1/3]

```
template<typename _Tp >
complex<_Tp> std::operator* (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value  $x$  times  $y$ .

Definition at line 386 of file complex.

**3.20.2.15 operator\*()** [2/3]

```
template<typename _Tp >
complex<_Tp> std::operator* (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value  $x$  times  $y$ .

Definition at line 395 of file complex.

**3.20.2.16 operator\*()** [3/3]

```
template<typename _Tp >
complex<_Tp> std::operator* (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value  $x$  times  $y$ .

Definition at line 404 of file complex.

**3.20.2.17 operator\*=( )** [1/2]

```
template<typename _Tp >
complex< _Tp > & std::complex< _Tp >::operator*= (
    const _Tp & __t )
```

Multiply this complex number by a scalar.

Definition at line 245 of file complex.

**3.20.2.18 operator\*=( )** [2/2]

```
template<typename _Tp >
template<typename _Up >
complex< _Tp > & std::complex< _Tp >::operator*= (
    const complex< _Up > & __z )
```

Multiply this complex number by another.

Definition at line 299 of file complex.

**3.20.2.19 operator+( )** [1/4]

```
template<typename _Tp >
complex<_Tp> std::operator+ (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x plus y.

Definition at line 326 of file complex.

**3.20.2.20 operator+( )** [2/4]

```
template<typename _Tp >
complex<_Tp> std::operator+ (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x plus y.

Definition at line 335 of file complex.

**3.20.2.21 operator+( )** [3/4]

```
template<typename _Tp >
complex<_Tp> std::operator+ (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x plus y.

Definition at line 344 of file complex.

**3.20.2.22 operator+()** [ 4 / 4 ]

```
template<typename _Tp >
complex<_Tp> std::operator+ (
    const complex< _Tp > & __x ) [inline]
```

Return x.

Definition at line 445 of file complex.

**3.20.2.23 operator+=()**

```
template<typename _Tp >
template<typename _Up >
complex< _Tp > & std::complex< _Tp >::operator+= (
    const complex< _Up > & __z )
```

Add another complex number to this one.

Definition at line 276 of file complex.

**3.20.2.24 operator-()** [ 1 / 4 ]

```
template<typename _Tp >
complex<_Tp> std::operator- (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x minus y.

Definition at line 356 of file complex.

**3.20.2.25 operator-()** [ 2 / 4 ]

```
template<typename _Tp >
complex<_Tp> std::operator- (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x minus y.

Definition at line 365 of file complex.

**3.20.2.26 operator-()** [3/4]

```
template<typename _Tp >
complex<_Tp> std::operator- (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value  $x$  minus  $y$ .

Definition at line 374 of file complex.

**3.20.2.27 operator-()** [4/4]

```
template<typename _Tp >
complex<_Tp> std::operator- (
    const complex< _Tp > & __x ) [inline]
```

Return complex negation of  $x$ .

Definition at line 451 of file complex.

**3.20.2.28 operator-=( )**

```
template<typename _Tp >
template<typename _Up >
complex< _Tp > & std::complex< _Tp >::operator-= (
    const complex< _Up > & __z )
```

Subtract another complex number from this one.

Definition at line 287 of file complex.

**3.20.2.29 operator/()** [1/3]

```
template<typename _Tp >
complex<_Tp> std::operator/ (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value  $x$  divided by  $y$ .

Definition at line 416 of file complex.

**3.20.2.30 operator/()** [2/3]

```
template<typename _Tp >
complex<_Tp> std::operator/ (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x divided by y.

Definition at line 425 of file complex.

**3.20.2.31 operator/()** [3/3]

```
template<typename _Tp >
complex<_Tp> std::operator/ (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x divided by y.

Definition at line 434 of file complex.

**3.20.2.32 operator/=()** [1/2]

```
template<typename _Tp >
complex< _Tp > & std::complex< _Tp >::operator/= (
    const _Tp & __t )
```

Divide this complex number by a scalar.

Definition at line 255 of file complex.

**3.20.2.33 operator/=()** [2/2]

```
template<typename _Tp >
template<typename _Up >
complex< _Tp > & std::complex< _Tp >::operator/= (
    const complex< _Up > & __z )
```

Divide this complex number by another.

Definition at line 312 of file complex.

**3.20.2.34 operator<<()**

```
template<typename _Tp , typename _CharT , class _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const complex< _Tp > & __x )
```

Insertion operator for complex values.

Definition at line 547 of file complex.

**3.20.2.35 operator=()** [1/2]

```
template<typename _Tp >
complex< _Tp > & std::complex< _Tp >::operator= (
    const _Tp & __t )
```

Assign a scalar to this complex number.

Definition at line 235 of file complex.

**3.20.2.36 operator=()** [2/2]

```
template<typename _Tp >
template<typename _Up >
complex< _Tp > & std::complex< _Tp >::operator= (
    const complex< _Up > & __z )
```

Assign another complex number to this one.

Definition at line 265 of file complex.

**3.20.2.37 operator==( )** [1/3]

```
template<typename _Tp >
constexpr bool std::operator==(
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return true if  $x$  is equal to  $y$ .

Definition at line 458 of file complex.



**3.20.2.38 operator==( ) [2/3]**

```
template<typename _Tp >
constexpr bool std::operator==(
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return true if *x* is equal to *y*.

Definition at line 463 of file `complex`.

**3.20.2.39 operator==( ) [3/3]**

```
template<typename _Tp >
constexpr bool std::operator==(
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return true if *x* is equal to *y*.

Definition at line 468 of file `complex`.

**3.20.2.40 operator>>( )**

```
template<typename _Tp , typename _CharT , class _Traits >
basic_istream<_CharT, _Traits>& std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    complex< _Tp > & __x )
```

Extraction operator for complex values.

Definition at line 493 of file `complex`.

**3.20.2.41 polar( )**

```
template<typename _Tp >
complex< _Tp > std::polar (
    const _Tp & __rho,
    const _Tp & __theta = 0 ) [inline]
```

Return complex with magnitude *rho* and angle *theta*.

Definition at line 690 of file `complex`.

**3.20.2.42 pow()** [1/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const complex< _Tp > & __z,
    int __n ) [inline]
```

Return  $x$  to the  $y$ 'th power.

Definition at line 1008 of file complex.

Referenced by `std::gamma_distribution< result_type >::operator>()`.

**3.20.2.43 pow()** [2/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const complex< _Tp > & __x,
    const _Tp & __y )
```

Return  $x$  to the  $y$ 'th power.

Definition at line 1017 of file complex.

**3.20.2.44 pow()** [3/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return  $x$  to the  $y$ 'th power.

Definition at line 1056 of file complex.

**3.20.2.45 pow()** [4/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return  $x$  to the  $y$ 'th power.

Definition at line 1062 of file complex.

**3.20.2.46 pow()** [5/5]

```
template<typename _Tp , typename _Up >
std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::pow (
    const std::complex< _Tp > & __x,
    const _Up & __y ) [inline]
```

Additional overloads [8.1.9].

Definition at line 350 of file tr1/complex.

**3.20.2.47 sin()**

```
template<typename _Tp >
complex< _Tp > std::sin (
    const complex< _Tp > & __z ) [inline]
```

Return complex sine of z.

Definition at line 848 of file complex.

**3.20.2.48 sinh()**

```
template<typename _Tp >
complex< _Tp > std::sinh (
    const complex< _Tp > & __z ) [inline]
```

Return complex hyperbolic sine of z.

Definition at line 878 of file complex.

**3.20.2.49 sqrt()**

```
template<typename _Tp >
complex< _Tp > std::sqrt (
    const complex< _Tp > & __z ) [inline]
```

Return complex square root of z.

Definition at line 922 of file complex.

Referenced by `std::normal_distribution< result_type >::operator()()`, and `std::student_t_distribution< _RealType >::operator()()`.

#### 3.20.2.50 tan()

```
template<typename _Tp >  
complex< _Tp > std::tan (  
    const complex< _Tp > & __z ) [inline]
```

Return complex tangent of z.

Definition at line 949 of file complex.

#### 3.20.2.51 tanh()

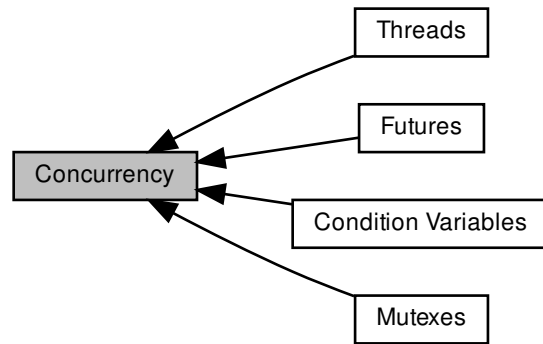
```
template<typename _Tp >  
complex< _Tp > std::tanh (  
    const complex< _Tp > & __z ) [inline]
```

Return complex hyperbolic tangent of z.

Definition at line 977 of file complex.

### 3.21 Concurrency

Collaboration diagram for Concurrency:



#### Modules

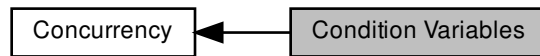
- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

#### 3.21.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

## 3.22 Condition Variables

Collaboration diagram for Condition Variables:



### Classes

- class `std::condition_variable`

### Enumerations

- enum `std::cv_status` { `no_timeout`, `timeout` }

### Functions

- void `std::notify_all_at_thread_exit` (`condition_variable` &, `unique_lock`<`mutex`>)

#### 3.22.1 Detailed Description

Classes for `condition_variable` support.

#### 3.22.2 Enumeration Type Documentation

##### 3.22.2.1 `cv_status`

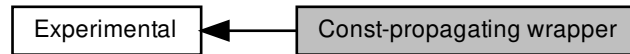
```
enum std::cv_status [strong]
```

`cv_status`

Definition at line 62 of file `condition_variable`.

### 3.23 Const-propagating wrapper

Collaboration diagram for Const-propagating wrapper:



#### Classes

- class `std::experimental::fundamentals_v2::propagate_const< _Tp >`

#### Functions

- `template<typename _Tp >`  
`constexpr const _Tp & std::experimental::fundamentals_v2::get_underlying (const propagate_const< _Tp >`  
`&__pt) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp & std::experimental::fundamentals_v2::get_underlying (propagate_const< _Tp > &__pt)`  
`noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt,`  
`nullptr_t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (nullptr_t, const propagate_const< _Tp >`  
`&__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, const`  
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const _Tp &__t, const propagate_const< ↵`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const`  
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const _Tp &__t, const propagate_const< ↵`  
`_Up > &__pu)`

- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate\_const< _Tp > &__pt,`  
`const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate\_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const _Tp &__t, const propagate\_const<`  
`_Up > &__pu)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate\_const< _Tp > &__pt,`  
`nullptr_t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (nullptr_t, const propagate\_const< _Tp >`  
`&__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate\_const< _Tp > &__pt,`  
`const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate\_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const _Tp &__t, const propagate\_const<`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate\_const< _Tp > &__pt, const`  
`propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate\_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const _Tp &__t, const propagate\_const< ←`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate\_const< _Tp > &__pt,`  
`const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate\_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const _Tp &__t, const propagate\_const<`  
`_Up > &__pu)`
- `template<typename _Tp >`  
`constexpr void std::experimental::fundamentals_v2::swap (propagate\_const< _Tp > &__pt, propagate\_const<`  
`_Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

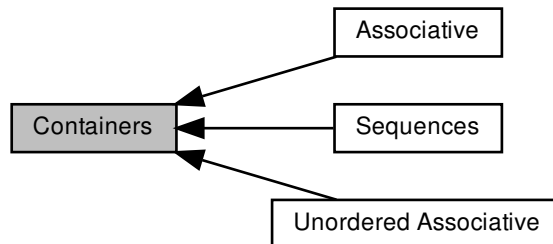
### 3.23.1 Detailed Description

A const-propagating wrapper that propagates const to pointer-like members, as described in n4388 "A Proposal to Add a Const-Propagating Wrapper to the Standard Library".



## 3.24 Containers

Collaboration diagram for Containers:



### Modules

- [Associative](#)
- [Sequences](#)
- [Unordered Associative](#)

#### 3.24.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

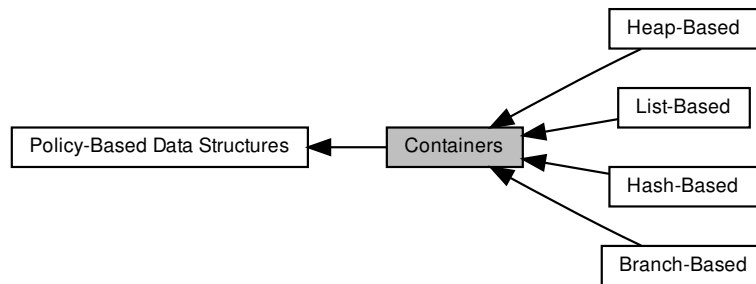
Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

## 3.25 Containers

Collaboration diagram for Containers:



### Modules

- [Branch-Based](#)
- [Hash-Based](#)
- [Heap-Based](#)
- [List-Based](#)

### 3.25.1 Detailed Description

### 3.26 Data Structure Type

Collaboration diagram for Data Structure Type:



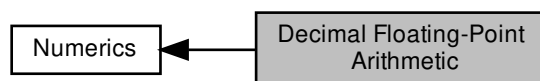
#### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_branch\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_tag](#)
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)

#### 3.26.1 Detailed Description

## 3.27 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



### Namespaces

- [std::decimal](#)

### 3.27.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

### 3.28 Diagnostics

Collaboration diagram for Diagnostics:



#### Modules

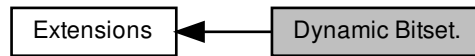
- [Exceptions](#)

#### 3.28.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

## 3.29 Dynamic Bitset.

Collaboration diagram for Dynamic Bitset.:



## Classes

- struct `std::tr2::__dynamic_bitset_base<_WordT, _Alloc>`
- class `std::tr2::dynamic_bitset<_WordT, _Alloc>`

## Functions

- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
void `std::tr2::dynamic_bitset<_WordT, _Alloc>::M_copy_to_string` (`std::basic_string<_CharT, _Traits, _Alloc1> &__str`, `_CharT __zero=_CharT('0')`, `_CharT __one=_CharT('1')`) const
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
`std::basic_ostream<_CharT, _Traits> & std::tr2::operator<<` (`std::basic_ostream<_CharT, _Traits> &__os`, const `dynamic_bitset<_WordT, _Alloc> &__x`)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
`std::basic_istream<_CharT, _Traits> & std::tr2::operator>>` (`std::basic_istream<_CharT, _Traits> &__is`, `dynamic_bitset<_WordT, _Alloc> &__x`)
- template<typename \_WordT, typename \_Alloc>  
bool `std::tr2::operator!=` (const `dynamic_bitset<_WordT, _Alloc> &__lhs`, const `dynamic_bitset<_WordT, _Alloc> &__rhs`)
- template<typename \_WordT, typename \_Alloc>  
bool `std::tr2::operator<=` (const `dynamic_bitset<_WordT, _Alloc> &__lhs`, const `dynamic_bitset<_WordT, _Alloc> &__rhs`)
- template<typename \_WordT, typename \_Alloc>  
bool `std::tr2::operator>` (const `dynamic_bitset<_WordT, _Alloc> &__lhs`, const `dynamic_bitset<_WordT, _Alloc> &__rhs`)
- template<typename \_WordT, typename \_Alloc>  
bool `std::tr2::operator>=` (const `dynamic_bitset<_WordT, _Alloc> &__lhs`, const `dynamic_bitset<_WordT, _Alloc> &__rhs`)
- template<typename \_WordT, typename \_Alloc>  
`dynamic_bitset<_WordT, _Alloc> std::tr2::operator&` (const `dynamic_bitset<_WordT, _Alloc> &__x`, const `dynamic_bitset<_WordT, _Alloc> &__y`)
- template<typename \_WordT, typename \_Alloc>  
`dynamic_bitset<_WordT, _Alloc> std::tr2::operator|` (const `dynamic_bitset<_WordT, _Alloc> &__x`, const `dynamic_bitset<_WordT, _Alloc> &__y`)

- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

### 3.29.1 Detailed Description

### 3.29.2 Function Documentation

#### 3.29.2.1 `operator!=()`

```
template<typename _WordT, typename _Alloc >
bool std::tr2::operator!= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1122 of file `dynamic_bitset`.

#### 3.29.2.2 `operator&()`

```
template<typename _WordT, typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator& (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

#### Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

#### Returns

A new bitset.

These should be self-explanatory.

Definition at line 1157 of file `dynamic_bitset`.

## 3.29.2.3 operator-()

```
template<typename _WordT , typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator- (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

## Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

## Returns

A new bitset.

These should be self-explanatory.

Definition at line 1187 of file `dynamic_bitset`.

## 3.29.2.4 operator&lt;&lt;()

```
template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
std::basic_ostream<_CharT, _Traits>& std::tr2::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const dynamic_bitset< _WordT, _Alloc > & __x ) [inline]
```

Stream output operator for `dynamic_bitset`.

Definition at line 1200 of file `dynamic_bitset`.

## 3.29.2.5 operator&lt;=()

```
template<typename _WordT , typename _Alloc >
bool std::tr2::operator<= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1128 of file `dynamic_bitset`.



### 3.29.2.6 operator>()

```
template<typename _WordT , typename _Alloc >
bool std::tr2::operator> (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1134 of file dynamic\_bitset.

### 3.29.2.7 operator>=()

```
template<typename _WordT , typename _Alloc >
bool std::tr2::operator>= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1140 of file dynamic\_bitset.

### 3.29.2.8 operator>>()

```
template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
std::basic_istream<_CharT, _Traits>& std::tr2::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    dynamic_bitset< _WordT, _Alloc > & __x )
```

Stream input operator for dynamic\_bitset.

Input will skip whitespace and only accept '0' and '1' characters. The dynamic\_bitset will grow as necessary to hold the string of bits.

Definition at line 207 of file dynamic\_bitset.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve(), std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::size(), and std::basic\_ios< \_CharT, \_Traits >::widen().

### 3.29.2.9 operator^()

```
template<typename _WordT , typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator^ (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

**Parameters**

$\_x$	A bitset.
$\_y$	A bitset of the same size as $\_x$ .

**Returns**

A new bitset.

These should be self-explanatory.

Definition at line 1177 of file dynamic\_bitset.

**3.29.2.10 operator" | ()**

```
template<typename _WordT , typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator| (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

**Parameters**

$\_x$	A bitset.
$\_y$	A bitset of the same size as $\_x$ .

**Returns**

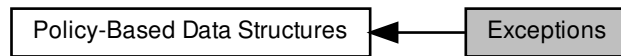
A new bitset.

These should be self-explanatory.

Definition at line 1167 of file dynamic\_bitset.

### 3.30 Exceptions

Collaboration diagram for Exceptions:



#### Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)
- struct [\\_\\_gnu\\_pbds::insert\\_error](#)
- struct [\\_\\_gnu\\_pbds::join\\_error](#)
- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

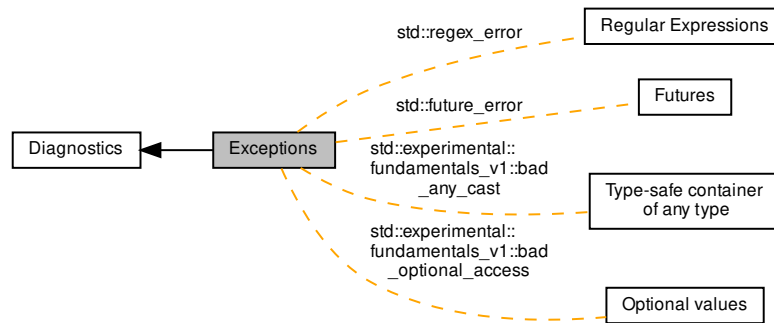
#### Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error\(\)](#)

#### 3.30.1 Detailed Description

## 3.31 Exceptions

Collaboration diagram for Exceptions:



## Classes

- class `__cxxabiv1::__forced_unwind`
- struct `__gnu_cxx::forced_error`
- class `__gnu_cxx::recursive_init_error`
- class `std::__exception_ptr::exception_ptr`
- class `std::bad_alloc`
- class `std::bad_cast`
- class `std::bad_function_call`
- class `std::bad_typeid`
- class `std::bad_weak_ptr`
- class `std::domain_error`
- class `std::exception`
- class `std::experimental::fundamentals_v1::bad_any_cast`
- class `std::experimental::fundamentals_v1::bad_optional_access`
- class `std::future_error`
- class `std::invalid_argument`
- class `std::ios_base::failure`
- class `std::length_error`
- class `std::logic_error`
- class `std::nested_exception`
- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `std::regex_error`
- class `std::runtime_error`
- class `std::system_error`
- class `std::underflow_error`

## Typedefs

- `template<typename _Tp >`  
`using std::__rethrow_if_nested_cond = typename enable_if< __and< is_polymorphic< _Tp >, __or< ↵`  
`__not< is_base_of< nested_exception, _Tp > >, is_convertible< _Tp *, nested_exception * > >>::value`  
`>::type`

## Functions

- `template<typename _Ex >`  
`__rethrow_if_nested_cond< _Ex > std::__rethrow_if_nested_impl (const _Ex * __ptr)`
- `void std::__rethrow_if_nested_impl (const void *)`
- `template<typename _Tp >`  
`void std::__throw_with_nested_impl (_Tp && __t, true_type)`
- `template<typename _Tp >`  
`void std::__throw_with_nested_impl (_Tp && __t, false_type)`
- `void __gnu_cxx::__verbose_terminate_handler ()`
- `exception_ptr std::current_exception () noexcept`
- `template<typename _Ex >`  
`exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void std::rethrow_if_nested (const _Ex & __ex)`
- `template<typename _Tp >`  
`void std::throw_with_nested (_Tp && __t)`
- `virtual const char * std::exception::what () const _GLIBCXX_TXN_SAFE_DYN noexcept`

### 3.31.1 Detailed Description

Classes and functions for reporting errors via exception classes.

### 3.31.2 Function Documentation

#### 3.31.2.1 \_\_verbose\_terminate\_handler()

```
void __gnu_cxx::__verbose_terminate_handler ( )
```

A replacement for the standard `terminate_handler` which prints more information about the terminating exception (if any) on `stderr`.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.↵html>

In 3.4 and later, this is on by default.

3.31.2.2 `current_exception()`

```
exception_ptr std::current_exception ( ) [noexcept]
```

Obtain an `exception_ptr` to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

3.31.2.3 `make_exception_ptr()`

```
template<typename _Ex >
exception_ptr std::make_exception_ptr (
    _Ex __ex ) [noexcept]
```

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 179 of file `exception_ptr.h`.

3.31.2.4 `rethrow_exception()`

```
void std::rethrow_exception (
    exception_ptr )
```

Throw the object pointed to by the `exception_ptr`.

3.31.2.5 `rethrow_if_nested()`

```
template<typename _Ex >
void std::rethrow_if_nested (
    const _Ex & __ex ) [inline]
```

If `__ex` is derived from `nested_exception`, `__ex.rethrow_nested()`.

Definition at line 151 of file `nested_exception.h`.

References `std::__addressof()`.

3.31.2.6 `throw_with_nested()`

```
template<typename _Tp >
void std::throw_with_nested (
    _Tp && __t ) [inline]
```

If `__t` is derived from `nested_exception`, throws `__t`. Else, throws an implementation-defined object derived from both.

Definition at line 114 of file `nested_exception.h`.

3.31.2.7 `what()`

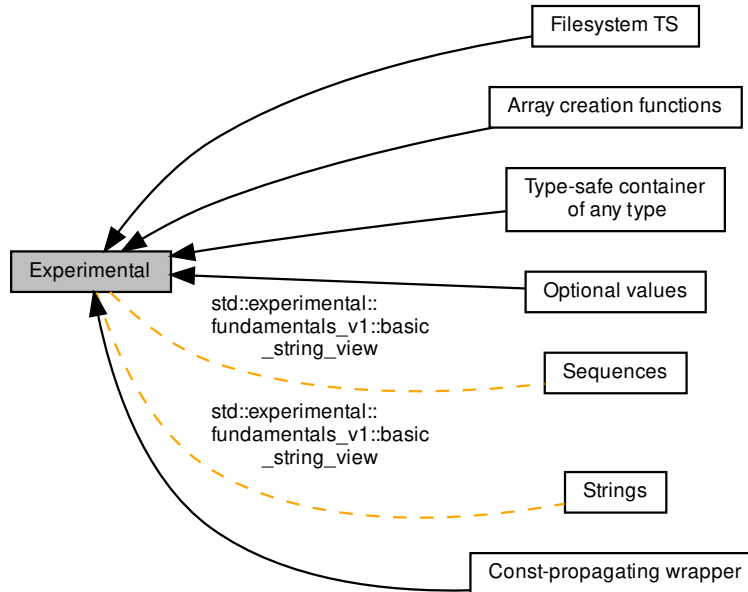
```
virtual const char* std::exception::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::ios_base::failure`, `std::runtime_error`, `std::bad_typeid`, `std::bad_cast`, `std::logic_error`, `std::future_error`, `std::bad_weak_ptr`, `std::experimental::fundamentals_v1::bad_any_cast`, `std::bad_alloc`, `std::bad_function_call`, and `std::bad_exception`.

### 3.32 Experimental

Collaboration diagram for Experimental:



#### Modules

- [Array creation functions](#)
- [Const-propagating wrapper](#)
- [Filesystem TS](#)
- [Optional values](#)
- [Type-safe container of any type](#)

#### Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view<\\_CharT, \\_Traits>](#)

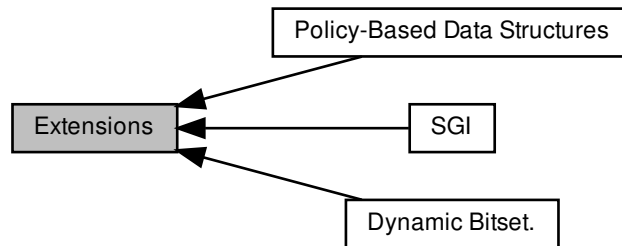
#### 3.32.1 Detailed Description

Components specified by various Technical Specifications.

As indicated by the `std::experimental` namespace and the header paths, the contents of these Technical Specifications are experimental and not part of the C++ standard. As such the interfaces and implementations may change in the future, and there is **no guarantee of compatibility between different GCC releases** for these features.

### 3.33 Extensions

Collaboration diagram for Extensions:



#### Modules

- [Dynamic Bitset.](#)
- [Policy-Based Data Structures](#)
- [SGI](#)

#### Classes

- [class `\_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base >`](#)

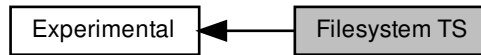
#### 3.33.1 Detailed Description

Components generally useful that are not part of any standard.



### 3.34 Filesystem TS

Collaboration diagram for Filesystem TS:



#### Classes

- class `std::experimental::filesystem::v1::path`

#### Typedefs

- using `std::experimental::filesystem::v1::file_time_type` = `std::chrono::system_clock::time_point`

#### Enumerations

- enum `std::experimental::filesystem::v1::copy_options` : unsigned short { **none**, **skip\_existing**, **overwrite\_existing**, **update\_existing**, **recursive**, **copy\_symlinks**, **skip\_symlinks**, **directories\_only**, **create\_symlinks**, **create\_hard\_links** }
- enum `directory_options` : unsigned char { **none**, **follow\_directory\_symlink**, **skip\_permission\_denied** }
- enum `file_type` : signed char { **none**, **not\_found**, **regular**, **directory**, **symlink**, **block**, **character**, **fifo**, **socket**, **unknown** }
- enum `std::experimental::filesystem::v1::perms` : unsigned { **none**, **owner\_read**, **owner\_write**, **owner\_exec**, **owner\_all**, **group\_read**, **group\_write**, **group\_exec**, **group\_all**, **others\_read**, **others\_write**, **others\_exec**, **others\_all**, **all**, **set\_uid**, **set\_gid**, **sticky\_bit**, **mask**, **unknown**, **add\_perms**, **remove\_perms**, **symlink\_nofollow** }

## Functions

- constexpr `copy_options` `std::experimental::filesystem::v1::operator& (copy_options __x, copy_options __y)` noexcept
- constexpr `perms` `std::experimental::filesystem::v1::operator& (perms __x, perms __y)` noexcept
- constexpr `directory_options` `std::experimental::filesystem::v1::operator& (directory_options __x, directory_options __y)` noexcept
- `copy_options` & `std::experimental::filesystem::v1::operator&= (copy_options &__x, copy_options __y)` noexcept
- `perms` & `std::experimental::filesystem::v1::operator&= (perms &__x, perms __y)` noexcept
- `directory_options` & `std::experimental::filesystem::v1::operator&= (directory_options &__x, directory_options __y)` noexcept
- constexpr `copy_options` `std::experimental::filesystem::v1::operator^ (copy_options __x, copy_options __y)` noexcept
- constexpr `perms` `std::experimental::filesystem::v1::operator^ (perms __x, perms __y)` noexcept
- constexpr `directory_options` `std::experimental::filesystem::v1::operator^ (directory_options __x, directory_options __y)` noexcept
- `copy_options` & `std::experimental::filesystem::v1::operator^= (copy_options &__x, copy_options __y)` noexcept
- `perms` & `std::experimental::filesystem::v1::operator^= (perms &__x, perms __y)` noexcept
- `directory_options` & `std::experimental::filesystem::v1::operator^= (directory_options &__x, directory_options __y)` noexcept
- constexpr `copy_options` `std::experimental::filesystem::v1::operator| (copy_options __x, copy_options __y)` noexcept
- constexpr `perms` `std::experimental::filesystem::v1::operator| (perms __x, perms __y)` noexcept
- constexpr `directory_options` `std::experimental::filesystem::v1::operator| (directory_options __x, directory_options __y)` noexcept
- `copy_options` & `std::experimental::filesystem::v1::operator|= (copy_options &__x, copy_options __y)` noexcept
- `perms` & `std::experimental::filesystem::v1::operator|= (perms &__x, perms __y)` noexcept
- `directory_options` & `std::experimental::filesystem::v1::operator|= (directory_options &__x, directory_options __y)` noexcept
- constexpr `copy_options` `std::experimental::filesystem::v1::operator~ (copy_options __x)` noexcept
- constexpr `perms` `std::experimental::filesystem::v1::operator~ (perms __x)` noexcept
- constexpr `directory_options` `std::experimental::filesystem::v1::operator~ (directory_options __x)` noexcept
- void `std::experimental::filesystem::v1::copy (const path &__from, const path &__to, copy_options __options)`
- void `std::experimental::filesystem::v1::copy (const path &__from, const path &__to, copy_options __options, error_code &)` noexcept
- bool `std::experimental::filesystem::v1::copy_file (const path &__from, const path &__to, copy_options __option)`
- bool `std::experimental::filesystem::v1::copy_file (const path &__from, const path &__to, copy_options __option, error_code &)` noexcept
- `path` `std::experimental::filesystem::v1::current_path ()`
- `file_status` `std::experimental::filesystem::v1::status (const path &)`
- `file_status` `std::experimental::filesystem::v1::status (const path &, error_code &)` noexcept
- bool `std::experimental::filesystem::v1::status_known (file_status)` noexcept
- `file_status` `std::experimental::filesystem::v1::symlink_status (const path &)`
- `file_status` `std::experimental::filesystem::v1::symlink_status (const path &, error_code &)` noexcept
- bool `std::experimental::filesystem::v1::is_regular_file (file_status)` noexcept
- bool `std::experimental::filesystem::v1::is_symlink (file_status)` noexcept

- `path std::experimental::filesystem::v1::absolute` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::experimental::filesystem::v1::canonical` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::experimental::filesystem::v1::canonical` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path std::experimental::filesystem::v1::canonical` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `void std::experimental::filesystem::v1::copy` (const `path` &\_\_from, const `path` &\_\_to)
- `void std::experimental::filesystem::v1::copy` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::copy_file` (const `path` &\_\_from, const `path` &\_\_to)
- `bool std::experimental::filesystem::v1::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::v1::copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new←\_symlink)
- `void std::experimental::filesystem::v1::copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new←\_symlink, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::create_directories` (const `path` &\_\_p)
- `bool std::experimental::filesystem::v1::create_directories` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &\_\_p)
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &\_\_p, const `path` &attributes)
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &\_\_p, const `path` &attributes, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::v1::create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new←\_symlink)
- `void std::experimental::filesystem::v1::create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new←\_symlink, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::v1::create_hard_link` (const `path` &\_\_to, const `path` &\_\_new\_hard\_link)
- `void std::experimental::filesystem::v1::create_hard_link` (const `path` &\_\_to, const `path` &\_\_new\_hard\_link, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::v1::create_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink)
- `void std::experimental::filesystem::v1::create_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- `path std::experimental::filesystem::v1::current_path` (`error_code` &\_\_ec)
- `void std::experimental::filesystem::v1::current_path` (const `path` &\_\_p)
- `void std::experimental::filesystem::v1::current_path` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::equivalent` (const `path` &\_\_p1, const `path` &\_\_p2)
- `bool std::experimental::filesystem::v1::equivalent` (const `path` &\_\_p1, const `path` &\_\_p2, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::exists` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::v1::exists` (const `path` &\_\_p)
- `bool std::experimental::filesystem::v1::exists` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `uintmax_t std::experimental::filesystem::v1::file_size` (const `path` &\_\_p)
- `uintmax_t std::experimental::filesystem::v1::file_size` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `uintmax_t std::experimental::filesystem::v1::hard_link_count` (const `path` &\_\_p)
- `uintmax_t std::experimental::filesystem::v1::hard_link_count` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::is_block_file` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::v1::is_block_file` (const `path` &\_\_p)
- `bool std::experimental::filesystem::v1::is_block_file` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::is_character_file` (file\_status \_\_s) noexcept
- `bool std::experimental::filesystem::v1::is_character_file` (const `path` &\_\_p)
- `bool std::experimental::filesystem::v1::is_character_file` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::v1::is_directory` (file\_status \_\_s) noexcept

- `bool std::experimental::filesystem::v1::is_directory (const path &__p)`
- `bool std::experimental::filesystem::v1::is_directory (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::is_empty (const path &__p)`
- `bool std::experimental::filesystem::v1::is_empty (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::is_fifo (file_status __s) noexcept`
- `bool std::experimental::filesystem::v1::is_fifo (const path &__p)`
- `bool std::experimental::filesystem::v1::is_fifo (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::is_other (file_status __s) noexcept`
- `bool std::experimental::filesystem::v1::is_other (const path &__p)`
- `bool std::experimental::filesystem::v1::is_other (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::is_regular_file (const path &__p)`
- `bool std::experimental::filesystem::v1::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::is_socket (file_status __s) noexcept`
- `bool std::experimental::filesystem::v1::is_socket (const path &__p)`
- `bool std::experimental::filesystem::v1::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::is_symlink (const path &__p)`
- `bool std::experimental::filesystem::v1::is_symlink (const path &__p, error_code &__ec) noexcept`
- `file_time_type std::experimental::filesystem::v1::last_write_time (const path &__p)`
- `file_time_type std::experimental::filesystem::v1::last_write_time (const path &__p, error_code &__ec) noexcept`
- `void std::experimental::filesystem::v1::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::experimental::filesystem::v1::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `void std::experimental::filesystem::v1::permissions (const path &__p, perms __prms)`
- `void std::experimental::filesystem::v1::permissions (const path &__p, perms __prms, error_code &__ec) noexcept`
- `path std::experimental::filesystem::v1::read_symlink (const path &__p)`
- `path std::experimental::filesystem::v1::read_symlink (const path &__p, error_code &__ec)`
- `bool std::experimental::filesystem::v1::remove (const path &__p)`
- `bool std::experimental::filesystem::v1::remove (const path &__p, error_code &__ec) noexcept`
- `uintmax_t std::experimental::filesystem::v1::remove_all (const path &__p)`
- `uintmax_t std::experimental::filesystem::v1::remove_all (const path &__p, error_code &__ec) noexcept`
- `void std::experimental::filesystem::v1::rename (const path &__from, const path &__to)`
- `void std::experimental::filesystem::v1::rename (const path &__from, const path &__to, error_code &__ec) noexcept`
- `void std::experimental::filesystem::v1::resize_file (const path &__p, uintmax_t __size)`
- `void std::experimental::filesystem::v1::resize_file (const path &__p, uintmax_t __size, error_code &__ec) noexcept`
- `space_info std::experimental::filesystem::v1::space (const path &__p)`
- `space_info std::experimental::filesystem::v1::space (const path &__p, error_code &__ec) noexcept`
- `path std::experimental::filesystem::v1::system_complete (const path &__p)`
- `path std::experimental::filesystem::v1::system_complete (const path &__p, error_code &__ec)`
- `path std::experimental::filesystem::v1::temp_directory_path ()`
- `path std::experimental::filesystem::v1::temp_directory_path (error_code &__ec)`

#### 3.34.1 Detailed Description

Utilities for performing operations on file systems and their components, such as paths, regular files, and directories.

### 3.34.2 Enumeration Type Documentation

#### 3.34.2.1 `copy_options`

```
enum std::experimental::filesystem::v1::copy_options : unsigned short [strong]
```

Bitmask type.

Definition at line 87 of file `experimental/bits/fs_fwd.h`.

#### 3.34.2.2 `perms`

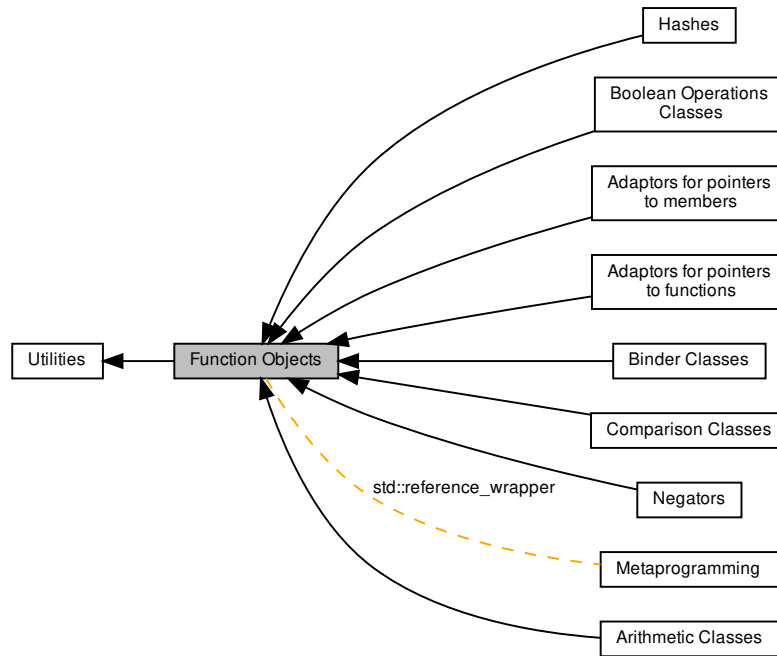
```
enum std::experimental::filesystem::v1::perms : unsigned [strong]
```

Bitmask type.

Definition at line 140 of file `experimental/bits/fs_fwd.h`.

## 3.35 Function Objects

Collaboration diagram for Function Objects:



## Modules

- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)
- [Arithmetic Classes](#)
- [Binder Classes](#)
- [Boolean Operations Classes](#)
- [Comparison Classes](#)
- [Hashes](#)
- [Negators](#)

## Classes

- `struct std::binary\_function< _Arg1, _Arg2, _Result >`
- `class std::function< _Res(_ArgTypes...)>`
- `class std::reference\_wrapper< _Tp >`
- `struct std::unary\_function< _Arg, _Result >`

## Functions

- `template<typename _Tp, typename _Class >  
_Mem_fn<_Tp _Class::*> std::mem\_fn (_Tp _Class::*__pm) noexcept`

### 3.35.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform(a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

### 3.35.2 Function Documentation

#### 3.35.2.1 `mem_fn()`

```
template<typename _Tp, typename _Class >  
_Mem_fn<_Tp _Class::*> std::mem_fn (  
    _Tp _Class::* __pm ) [inline], [noexcept]
```

Returns a function object that forwards to the member pointer *pm*.

Definition at line 160 of file `functional`.

### 3.36 Futures

Collaboration diagram for Futures:



#### Classes

- class `std::__basic_future< _Res >`
- struct `std::__future_base`
- struct `std::__future_base::Result< _Res & >`
- struct `std::__future_base::Result< void >`
- class `std::future< _Res >`
- class `std::future< _Res & >`
- class `std::future< void >`
- class `std::future_error`
- struct `std::is_error_code_enum< future_errc >`
- class `std::packaged_task< _Res(_ArgTypes...)>`
- class `std::promise< _Res >`
- class `std::promise< _Res & >`
- class `std::promise< void >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res & >`
- class `std::shared_future< void >`

#### Typedefs

- template<typename \_Fn, typename... \_Args>  
using `std::__async_result_of` = typename `result_of< typename decay< _Fn >::type(typename decay< _Args >::type...)>::type`

#### Enumerations

- enum `std::future_errc` { `future_already_retrieved`, `promise_already_satisfied`, `no_state`, `broken_promise` }
- enum `std::future_status` { `ready`, `timeout`, `deferred` }
- enum `std::launch` { `async`, `deferred` }



## Functions

- `std::__basic_future< _Res >::__basic_future` (const `shared_future< _Res >` &) noexcept
- `std::__basic_future< _Res >::__basic_future` (`shared_future< _Res >` &&) noexcept
- `std::__basic_future< _Res >::__basic_future` (`future< _Res >` &&) noexcept
- `template<typename _Signature, typename _Fn, typename _Alloc >`  
`static shared_ptr< __future_base::__Task_state_base< _Signature > > std::__create_task_state` (`_Fn` && `__fn`,  
`const _Alloc` & `__a`)
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > std::__future_base::__S_make_async_state` (`_BoundFn` && `__fn`)
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > std::__future_base::__S_make_deferred_state` (`_BoundFn` && `__fn`)
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > std::async` (`launch` `__policy`, `_Fn` && `__fn`, `_Args` &&... `__args`)
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > std::async` (`_Fn` && `__fn`, `_Args` &&... `__args`)
- `const error_category & std::future_category` () noexcept
- `error_code std::make_error_code` (`future_errc` `__errc`) noexcept
- `error_condition std::make_error_condition` (`future_errc` `__errc`) noexcept
- `constexpr launch std::operator&` (`launch` `__x`, `launch` `__y`)
- `launch & std::operator&=` (`launch` & `__x`, `launch` `__y`)
- `constexpr launch std::operator^` (`launch` `__x`, `launch` `__y`)
- `launch & std::operator^=` (`launch` & `__x`, `launch` `__y`)
- `constexpr launch std::operator|` (`launch` `__x`, `launch` `__y`)
- `launch & std::operator|=` (`launch` & `__x`, `launch` `__y`)
- `constexpr launch std::operator~` (`launch` `__x`)
- `shared_future< _Res > std::future< _Res >::share` () noexcept
- `shared_future< _Res & > std::future< _Res & >::share` () noexcept
- `shared_future< void > std::future< void >::share` () noexcept
- `template<typename _Res >`  
`void std::swap` (`promise< _Res >` & `__x`, `promise< _Res >` & `__y`) noexcept
- `template<typename _Res, typename... _ArgTypes>`  
`void std::swap` (`packaged_task< _Res(_ArgTypes...) >` & `__x`, `packaged_task< _Res(_ArgTypes...) >` & `__y`) noexcept

### 3.36.1 Detailed Description

Classes for futures support.

### 3.36.2 Enumeration Type Documentation

#### 3.36.2.1 future\_errc

```
enum std::future_errc [strong]
```

Error code for futures.

Definition at line 66 of file future.

### 3.36.2.2 future\_status

```
enum std::future_status [strong]
```

Status code for futures.

Definition at line 174 of file future.

### 3.36.2.3 launch

```
enum std::launch [strong]
```

Launch code for futures.

Definition at line 137 of file future.

## 3.36.3 Function Documentation

### 3.36.3.1 async() [1/2]

```
template<typename _Fn , typename... _Args>  
future< __async_result_of< _Fn, _Args... > > std::async (   
    launch __policy,  
    _Fn && __fn,  
    _Args &&... __args )
```

async

Definition at line 1712 of file future.

### 3.36.3.2 async() [2/2]

```
template<typename _Fn , typename... _Args>  
future< __async_result_of< _Fn, _Args... > > std::async (   
    _Fn && __fn,  
    _Args &&... __args ) [inline]
```

async, potential overload

Definition at line 1745 of file future.

### 3.36.3.3 future\_category()

```
const error_category& std::future_category ( ) [noexcept]
```

Points to a statically-allocated object derived from error\_category.

### 3.36.3.4 make\_error\_code()

```
error_code std::make_error_code (
    future_errc __errc ) [inline], [noexcept]
```

Overload for make\_error\_code.

Definition at line 84 of file future.

### 3.36.3.5 make\_error\_condition()

```
error_condition std::make_error_condition (
    future_errc __errc ) [inline], [noexcept]
```

Overload for make\_error\_condition.

Definition at line 89 of file future.

### 3.36.3.6 swap()

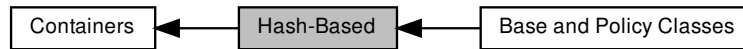
```
template<typename _Res , typename... _ArgTypes>
void std::swap (
    packaged_task< _Res(_ArgTypes...)> & __x,
    packaged_task< _Res(_ArgTypes...)> & __y ) [inline], [noexcept]
```

swap

Definition at line 1579 of file future.

### 3.37 Hash-Based

Collaboration diagram for Hash-Based:



#### Modules

- [Base and Policy Classes](#)

#### Classes

- [class `\_\_gnu\_pbds::basic\_hash\_table`](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc >
- [class `\_\_gnu\_pbds::cc\_hash\_table`](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- [class `\_\_gnu\_pbds::gp\_hash\_table`](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >

#### Macros

- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`

#### 3.37.1 Detailed Description

### 3.38 Hashes

Collaboration diagram for Hashes:



#### Classes

- struct `std::hash< _Tp >`
- struct `std::hash< _Tp * >`
- struct `std::hash< bool >`
- struct `std::hash< char >`
- struct `std::hash< char16_t >`
- struct `std::hash< char32_t >`
- struct `std::hash< double >`
- struct `std::hash< float >`
- struct `std::hash< int >`
- struct `std::hash< long >`
- struct `std::hash< long double >`
- struct `std::hash< long long >`
- struct `std::hash< short >`
- struct `std::hash< signed char >`
- struct `std::hash< unsigned char >`
- struct `std::hash< unsigned int >`
- struct `std::hash< unsigned long >`
- struct `std::hash< unsigned long long >`
- struct `std::hash< unsigned short >`
- struct `std::hash< wchar_t >`

#### Macros

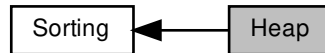
- `#define Cxx_hashtable_define_trivial_hash(_Tp)`

#### 3.38.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

### 3.39 Heap

Collaboration diagram for Heap:



#### Functions

- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

#### 3.39.1 Detailed Description

#### 3.39.2 Function Documentation

**3.39.2.1 is\_heap()** [1/2]

```
template<typename _RandomAccessIterator >
bool std::is_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Determines whether a range is a heap.

**Parameters**

<i>__first</i>	Start of range.
<i>__last</i>	End of range.

**Returns**

True if range is a heap, false otherwise.

Definition at line 529 of file `stl_heap.h`.

References `std::is_heap_until()`.

**3.39.2.2 is\_heap()** [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
bool std::is_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Determines whether a range is a heap using comparison functor.

**Parameters**

<i>__first</i>	Start of range.
<i>__last</i>	End of range.
<i>__comp</i>	Comparison functor to use.

**Returns**

True if range is a heap, false otherwise.

Definition at line 542 of file `stl_heap.h`.

3.39.2.3 `is_heap_until()` [1/2]

```
template<typename _RandomAccessIterator >
_RandomAccessIterator std::is_heap_until (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Search the end of a heap.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

## Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is a heap.

Definition at line 477 of file `stl_heap.h`.

3.39.2.4 `is_heap_until()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::is_heap_until (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Search the end of a heap using comparison functor.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

## Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is a heap. Comparisons are made using `__comp`.

Definition at line 505 of file `stl_heap.h`.

Referenced by `std::is_heap()`.



### 3.39.2.5 make\_heap() [1/2]

```
template<typename _RandomAccessIterator >
void std::make_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Construct a heap over a range.

#### Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation makes the elements in [`__first`,`__last`) into a heap.

Definition at line 360 of file `stl_heap.h`.

### 3.39.2.6 make\_heap() [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
void std::make_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Construct a heap over a range using comparison functor.

#### Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation makes the elements in [`__first`,`__last`) into a heap. Comparisons are made using `__comp`.

Definition at line 386 of file `stl_heap.h`.

### 3.39.2.7 pop\_heap() [1/2]

```
template<typename _RandomAccessIterator >
void std::pop_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Pop an element off a heap.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

## Precondition

`[__first, __last)` is a valid, non-empty range.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap.

Definition at line 271 of file `stl_heap.h`.

3.39.2.8 `pop_heap()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare >
void std::pop_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Pop an element off a heap using comparison functor.

## Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap. Comparisons are made using `comp`.

Definition at line 304 of file `stl_heap.h`.

3.39.2.9 `push_heap()` [1/2]

```
template<typename _RandomAccessIterator >
void std::push_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Push an element onto a heap.

**Parameters**

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.

This operation pushes the element at `last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap.

Definition at line 154 of file `stl_heap.h`.

**3.39.2.10 `push_heap()`** [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
void std::push_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Push an element onto a heap using comparison functor.

**Parameters**

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.
<code>__comp</code>	Comparison functor.

This operation pushes the element at `__last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap. Compare operations are performed using `comp`.

Definition at line 189 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::push()`.

**3.39.2.11 `sort_heap()`** [1/2]

```
template<typename _RandomAccessIterator >
void std::sort_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Sort a heap.

**Parameters**

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation sorts the valid heap in the range [`__first`,`__last`).

Definition at line 422 of file `stl_heap.h`.

#### 3.39.2.12 `sort_heap()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
void std::sort_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort a heap using comparison functor.

##### Parameters

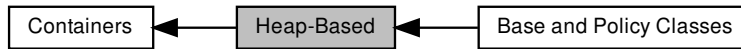
<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation sorts the valid heap in the range [`__first`,`__last`). Comparisons are made using `__comp`.

Definition at line 449 of file `stl_heap.h`.

### 3.40 Heap-Based

Collaboration diagram for Heap-Based:



#### Modules

- [Base and Policy Classes](#)

#### Classes

- class [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>](#)

#### Typedefs

- typedef `_Alloc` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::allocator\\_type](#)
- typedef `Cmp_Fn` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::cmp\\_fn](#)
- typedef `base_type::const_iterator` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::const\\_iterator](#)
- typedef `__rebind_va::const_pointer` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::const\\_pointer](#)
- typedef `__rebind_va::const_reference` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::const\\_reference](#)
- typedef `Tag` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::container\\_category](#)
- typedef `allocator_type::difference_type` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::difference\\_type](#)
- typedef `base_type::iterator` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::iterator](#)
- typedef `base_type::point_const_iterator` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::point\\_const\\_iterator](#)
- typedef `base_type::point_iterator` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::point\\_iterator](#)
- typedef `__rebind_va::pointer` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::pointer](#)
- typedef `__rebind_va::reference` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::reference](#)
- typedef `allocator_type::size_type` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::size\\_type](#)
- typedef `_Tv` [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>::value\\_type](#)

## Functions

- `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (const cmp\_fn &r\_cmp\_fn)
- `template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (It first\_it, It last\_it)
- `template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (It first\_it, It last\_it, const cmp\_fn &r\_cmp\_fn)
- `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (const `priority_queue` &other)
- `priority_queue` & `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::operator=` (const `priority_queue` &other)
- `void __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::swap` (`priority_queue` &other)

## 3.40.1 Detailed Description

## 3.40.2 Function Documentation

3.40.2.1 `priority_queue()` [1/3]

```
template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue (
    const cmp_fn & r_cmp_fn ) [inline]
```

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

Definition at line 116 of file `priority_queue.hpp`.

3.40.2.2 `priority_queue()` [2/3]

```
template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue (
    It first_it,
    It last_it ) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 122 of file `priority_queue.hpp`.

### 3.40.2.3 `priority_queue()` [3/3]

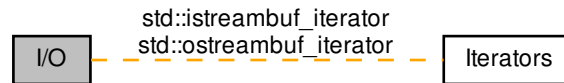
```
template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>>
template<typename It >
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
    It first_it,
    It last_it,
    const cmp_fn & r_cmp_fn ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object.

Definition at line 130 of file `priority_queue.hpp`.

## 3.41 I/O

Collaboration diagram for I/O:



## Classes

- class `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`
- class `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`
- class `std::basic_filebuf< _CharT, _Traits >`
- class `std::basic_fstream< _CharT, _Traits >`
- class `std::basic_ifstream< _CharT, _Traits >`
- class `std::basic_ios< _CharT, _Traits >`
- class `std::basic_iostream< _CharT, _Traits >`
- class `std::basic_istream< _CharT, _Traits >`
- class `std::basic_istreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_ofstream< _CharT, _Traits >`
- class `std::basic_ostream< _CharT, _Traits >`
- class `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_streambuf< _CharT, _Traits >`
- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`
- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`
- class `std::ios_base`
- class `std::istreambuf_iterator< _CharT, _Traits >`
- class `std::ostreambuf_iterator< _CharT, _Traits >`

## Typedefs

- typedef `basic_filebuf< char >` `std::filebuf`
- typedef `basic_fstream< char >` `std::fstream`
- typedef `basic_ifstream< char >` `std::ifstream`
- typedef `basic_ios< char >` `std::ios`
- typedef `basic_iostream< char >` `std::iostream`
- typedef `basic_istream< char >` `std::istream`
- typedef `basic_istreamstream< char >` `std::istreamstream`
- typedef `basic_ofstream< char >` `std::ofstream`
- typedef `basic_ostream< char >` `std::ostream`
- typedef `basic_ostreamstream< char >` `std::ostreamstream`
- typedef `basic_streambuf< char >` `std::streambuf`



- typedef `basic_stringbuf`< char > `std::stringbuf`
- typedef `basic_stringstream`< char > `std::stringstream`
- typedef `basic_filebuf`< wchar\_t > `std::wfilebuf`
- typedef `basic_fstream`< wchar\_t > `std::wfstream`
- typedef `basic_ifstream`< wchar\_t > `std::wifstream`
- typedef `basic_ios`< wchar\_t > `std::wios`
- typedef `basic_iostream`< wchar\_t > `std::wiostream`
- typedef `basic_istream`< wchar\_t > `std::wistream`
- typedef `basic_istreamstream`< wchar\_t > `std::wistreamstream`
- typedef `basic_ofstream`< wchar\_t > `std::wofstream`
- typedef `basic_ostream`< wchar\_t > `std::wostream`
- typedef `basic_ostreamstream`< wchar\_t > `std::wostreamstream`
- typedef `basic_streambuf`< wchar\_t > `std::wstreambuf`
- typedef `basic_stringbuf`< wchar\_t > `std::wstringbuf`
- typedef `basic_stringstream`< wchar\_t > `std::wstringstream`

### 3.41.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html#std.io.objects>

### 3.41.2 Typedef Documentation

#### 3.41.2.1 filebuf

```
typedef basic_filebuf<char> std::filebuf
```

Class for `char` file buffers.

Definition at line 159 of file `iosfwd`.

#### 3.41.2.2 fstream

```
typedef basic_fstream<char> std::fstream
```

Class for `char` mixed input and output file streams.

Definition at line 168 of file `iosfwd`.

#### 3.41.2.3 ifstream

```
typedef basic_ifstream<char> std::ifstream
```

Class for `char` input file streams.

Definition at line 162 of file `iosfwd`.

#### 3.41.2.4 ios

```
typedef basic_ios<char> std::ios
```

Base class for `char` streams.

Definition at line 128 of file `iosfwd`.

#### 3.41.2.5 iostream

```
typedef basic_iostream<char> std::iostream
```

Base class for `char` mixed input and output streams.

Definition at line 144 of file `iosfwd`.

#### 3.41.2.6 istream

```
typedef basic_istream<char> std::istream
```

Base class for `char` input streams.

Definition at line 138 of file `iosfwd`.

#### 3.41.2.7 `istringstream`

```
typedef basic_istringstream<char> std::istringstream
```

Class for `char` input memory streams.

Definition at line 150 of file `iosfwd`.

#### 3.41.2.8 `ofstream`

```
typedef basic_ofstream<char> std::ofstream
```

Class for `char` output file streams.

Definition at line 165 of file `iosfwd`.

#### 3.41.2.9 `ostream`

```
typedef basic_ostream<char> std::ostream
```

Base class for `char` output streams.

Definition at line 141 of file `iosfwd`.

#### 3.41.2.10 `ostringstream`

```
typedef basic_ostringstream<char> std::ostringstream
```

Class for `char` output memory streams.

Definition at line 153 of file `iosfwd`.

#### 3.41.2.11 `streambuf`

```
typedef basic_streambuf<char> std::streambuf
```

Base class for `char` buffers.

Definition at line 135 of file `iosfwd`.

#### 3.41.2.12 `stringbuf`

```
typedef basic_stringbuf<char> std::stringbuf
```

Class for `char` memory buffers.

Definition at line 147 of file `iosfwd`.

#### 3.41.2.13 `stringstream`

```
typedef basic_stringstream<char> std::stringstream
```

Class for `char` mixed input and output memory streams.

Definition at line 156 of file `iosfwd`.

#### 3.41.2.14 `wfilebuf`

```
typedef basic_filebuf<wchar_t> std::wfilebuf
```

Class for `wchar_t` file buffers.

Definition at line 199 of file `iosfwd`.

#### 3.41.2.15 `wfstream`

```
typedef basic_fstream<wchar_t> std::wfstream
```

Class for `wchar_t` mixed input and output file streams.

Definition at line 208 of file `iosfwd`.

#### 3.41.2.16 `wifstream`

```
typedef basic_ifstream<wchar_t> std::wifstream
```

Class for `wchar_t` input file streams.

Definition at line 202 of file `iosfwd`.

#### 3.41.2.17 wios

```
typedef basic_ios<wchar_t> std::wios
```

Base class for `wchar_t` streams.

Definition at line 172 of file `iosfwd`.

#### 3.41.2.18 wiostream

```
typedef basic_iostream<wchar_t> std::wiostream
```

Base class for `wchar_t` mixed input and output streams.

Definition at line 184 of file `iosfwd`.

#### 3.41.2.19 wistream

```
typedef basic_istream<wchar_t> std::wistream
```

Base class for `wchar_t` input streams.

Definition at line 178 of file `iosfwd`.

#### 3.41.2.20 wstringstream

```
typedef basic_istringstream<wchar_t> std::wstringstream
```

Class for `wchar_t` input memory streams.

Definition at line 190 of file `iosfwd`.

#### 3.41.2.21 wofstream

```
typedef basic_ofstream<wchar_t> std::wofstream
```

Class for `wchar_t` output file streams.

Definition at line 205 of file `iosfwd`.

#### 3.41.2.22 wostream

```
typedef basic_ostream<wchar_t> std::wostream
```

Base class for `wchar_t` output streams.

Definition at line 181 of file `iosfwd`.

#### 3.41.2.23 wostringstream

```
typedef basic_ostringstream<wchar_t> std::wostringstream
```

Class for `wchar_t` output memory streams.

Definition at line 193 of file `iosfwd`.

#### 3.41.2.24 wstreambuf

```
typedef basic_streambuf<wchar_t> std::wstreambuf
```

Base class for `wchar_t` buffers.

Definition at line 175 of file `iosfwd`.

#### 3.41.2.25 wstringbuf

```
typedef basic_stringbuf<wchar_t> std::wstringbuf
```

Class for `wchar_t` memory buffers.

Definition at line 187 of file `iosfwd`.

#### 3.41.2.26 wstringstream

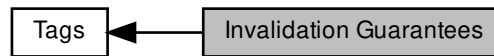
```
typedef basic_stringstream<wchar_t> std::wstringstream
```

Class for `wchar_t` mixed input and output memory streams.

Definition at line 196 of file `iosfwd`.

### 3.42 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



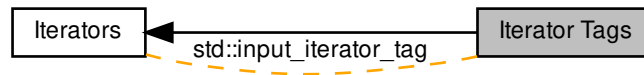
#### Classes

- struct [\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::range\\_invalidation\\_guarantee](#)

#### 3.42.1 Detailed Description

### 3.43 Iterator Tags

Collaboration diagram for Iterator Tags:



#### Classes

- struct `std::bidirectional_iterator_tag`
- struct `std::forward_iterator_tag`
- struct `std::input_iterator_tag`
- struct `std::output_iterator_tag`
- struct `std::random_access_iterator_tag`

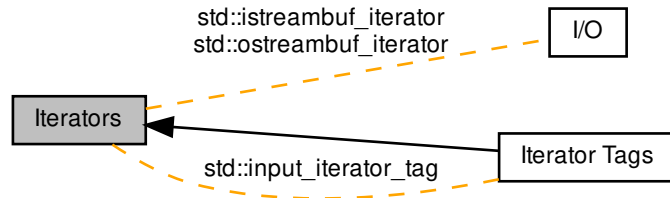
#### 3.43.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.



### 3.44 Iterators

Collaboration diagram for Iterators:



#### Modules

- [Iterator Tags](#)

#### Classes

- struct `std::__iterator_traits<_Iterator, typename>`
- class `std::back_insert_iterator<_Container>`
- class `std::front_insert_iterator<_Container>`
- struct `std::input_iterator_tag`
- class `std::insert_iterator<_Container>`
- class `std::istream_iterator<_Tp, _CharT, _Traits, _Dist>`
- class `std::istreambuf_iterator<_CharT, _Traits>`
- struct `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`
- struct `std::iterator_traits<_Tp*>`
- struct `std::iterator_traits<const _Tp*>`
- class `std::move_iterator<_Iterator>`
- class `std::ostream_iterator<_Tp, _CharT, _Traits>`
- class `std::ostreambuf_iterator<_CharT, _Traits>`
- class `std::reverse_iterator<_Iterator>`

#### Macros

- `#define __cpp_lib_make_reverse_iterator`

## Functions

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`  
`copy_move_a2 ( _CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`  
`copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__`  
`copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _Iter >`  
`constexpr iterator_traits< _Iter >::iterator_category std::__iterator_category (const _Iter &)`
- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond<typename iterator_traits<__`  
`_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`  
`_GLIBCXX17_CONSTEXPR _ReturnType std::__make_move_if_noexcept_iterator ( _Iterator __i)`
- `template<typename _Tp, typename _ReturnType = typename conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_`  
`__iterator<_Tp*>>::type>`  
`_GLIBCXX17_CONSTEXPR _ReturnType std::__make_move_if_noexcept_iterator ( _Tp * __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Iterator > std::__make_reverse_iterator ( _Iterator __i)`
- `template<typename _Iterator >`  
`auto std::__miter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__miter_`  
`base(__it.base()))`
- `template<typename _Iterator >`  
`auto std::__niter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__niter_`  
`base(__it.base()))`
- `template<typename _CharT, typename _Distance >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type std::advance (istreambuf_iterator<`  
`_CharT > & __i, _Distance __n)`
- `template<typename _Container >`  
`back_insert_iterator< _Container > std::back_inserter ( _Container & __x)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`  
`__result)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > std::front_inserter ( _Container & __x)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > std::inserter ( _Container & __x, _Iterator __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR move_iterator< _Iterator > std::make_move_iterator ( _Iterator __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Iterator > std::make_reverse_iterator ( _Iterator __i)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`  
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp,`  
`_CharT, _Traits, _Dist > & __y)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT,`  
`_Traits > & __b)`

- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const reverse\_iterator< _Iterator > &__x, const`  
`reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const reverse\_iterator< _IteratorL > &__x, const`  
`reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const move\_iterator< _IteratorL > &__x, const`  
`move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const move\_iterator< _Iterator > &__x, const move\_iterator<`  
`_Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse\_iterator< _Iterator > std::operator+ (typename reverse\_iterator< _Iterator >::difference_type __n, const reverse\_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR move\_iterator< _Iterator > std::operator+ (typename move\_iterator< _Iterator`  
`>::difference_type __n, const move\_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR auto std::operator- (const reverse\_iterator< _IteratorL > &__x, const`  
`reverse\_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR auto std::operator- (const move\_iterator< _IteratorL > &__x, const move\_iterator<`  
`_IteratorR > &__y) -> decltype(__x.base() - __y.base())`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator< (const reverse\_iterator< _Iterator > &__x, const`  
`reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator< (const reverse\_iterator< _IteratorL > &__x, const`  
`reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator<`  
`_IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator< (const move\_iterator< _Iterator > &__x, const move\_iterator<`  
`_Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<= (const reverse\_iterator< _Iterator > &__x, const`  
`reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<= (const reverse\_iterator< _IteratorL > &__x, const`  
`reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<= (const move\_iterator< _IteratorL > &__x, const`  
`move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<= (const move\_iterator< _Iterator > &__x, const`  
`move\_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool std::operator== (const istream\_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream\_iterator< _Tp,`  
`_CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf\_iterator< _CharT, _Traits > &__a, const istreambuf\_iterator< _CharT,`  
`_Traits > &__b)`

- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator== (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator== (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator== (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator== (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`

#### 3.44.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

#### 3.44.2 Function Documentation

### 3.44.2.1 `__iterator_category()`

```
template<typename _Iter >
constexpr iterator_traits<_Iter>::iterator_category std::__iterator_category (
    const _Iter & ) [inline]
```

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 205 of file `stl_iterator_base_types.h`.

Referenced by `std::advance()`, `std::distance()`, and `std::uninitialized_copy_n()`.

### 3.44.2.2 `back_inserter()`

```
template<typename _Container >
back_insert_iterator<_Container> std::back_inserter (
    _Container & __x ) [inline]
```

#### Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

#### Returns

An instance of `back_insert_iterator` working on `__x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 533 of file `bits/stl_iterator.h`.

Referenced by `std::match_results<_Bi_iter>::format()`, and `std::regex_replace()`.

### 3.44.2.3 `front_inserter()`

```
template<typename _Container >
front_insert_iterator<_Container> std::front_inserter (
    _Container & __x ) [inline]
```

#### Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

**Returns**

An instance of `front_insert_iterator` working on `x`.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 624 of file `bits/stl_iterator.h`.

**3.44.2.4 inserter()**

```
template<typename _Container , typename _Iterator >
insert_iterator<_Container> std::inserter (
    _Container & __x,
    _Iterator __i ) [inline]
```

**Parameters**

<code>__x</code>	A container of arbitrary type.
<code>__i</code>	An iterator into the container.

**Returns**

An instance of `insert_iterator` working on `__x`.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 739 of file `bits/stl_iterator.h`.

**3.44.2.5 make\_reverse\_iterator()**

```
template<typename _Iterator >
_GLIBCXX17_CONSTEXPR reverse_iterator<_Iterator> std::make_reverse_iterator (
    _Iterator __i ) [inline]
```

Generator function for `reverse_iterator`.

Definition at line 419 of file `bits/stl_iterator.h`.

#### 3.44.2.6 operator!=(())

```
template<class _Tp , class _CharT , class _Traits , class _Dist >
bool std::operator!=(
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y ) [inline]
```

Return false if x and y are both end or not end, or x and y are the same.

Definition at line 137 of file stream\_iterator.h.

#### 3.44.2.7 operator==( [1/2]

```
template<typename _Tp , typename _CharT , typename _Traits , typename _Dist >
bool std::operator==(
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y ) [inline]
```

Return true if x and y are both end or not end, or x and y are the same.

Definition at line 130 of file stream\_iterator.h.

#### 3.44.2.8 operator==( [2/2]

```
template<typename _Iterator >
_GLIBCXX17_CONSTEXPR bool std::operator==(
    const reverse_iterator< _Iterator > & __x,
    const reverse_iterator< _Iterator > & __y ) [inline]
```

##### Parameters

$\leftrightarrow$ __x	A reverse_iterator.
$\leftrightarrow$ __y	A reverse_iterator.

##### Returns

A simple bool.

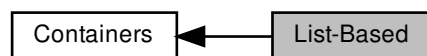
Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 302 of file bits/stl\_iterator.h.

References std::reverse\_iterator< \_Iterator >::base().

### 3.45 List-Based

Collaboration diagram for List-Based:



#### Classes

- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`

#### Macros

- `#define PB_DS_LU_BASE`

#### 3.45.1 Detailed Description



### 3.46 Locales

#### Classes

- class `std::codecvt< _InternT, _ExternT, _StateT >`
- class `std::ctype< _CharT >`
- class `std::ctype< char >`
- class `std::ctype< wchar_t >`
- class `std::locale`
- class `std::locale::facet`
- class `std::locale::id`
- class `std::messages< _CharT >`
- struct `std::messages_base`
- class `std::money_base`
- class `std::money_get< _CharT, _InIter >`
- class `std::money_put< _CharT, _OutIter >`
- class `std::moneypunct< _CharT, _Intl >`
- class `std::num_get< _CharT, _InIter >`
- class `std::num_put< _CharT, _OutIter >`
- class `std::numpunct< _CharT >`
- class `std::time_base`
- class `std::time_get< _CharT, _InIter >`
- class `std::time_put< _CharT, _OutIter >`
- class `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`
- class `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >`

#### Functions

- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn >`  
`bool std::__do_str_codecvt (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_in (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_in (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_out (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_out (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _Facet >`  
`bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use_facet (const locale &__loc)`

#### 3.46.1 Detailed Description

Classes and functions for internationalization and localization.

## 3.46.2 Function Documentation

## 3.46.2.1 has\_facet()

```
template<typename _Facet >
bool std::has_facet (
    const locale & __loc ) throw ( )
```

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

## Template Parameters

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

## Parameters

<code>__loc</code>	The locale to test.
--------------------	---------------------

## Returns

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file locale\_classes.tcc.

## 3.46.2.2 use\_facet()

```
template<typename _Facet >
const _Facet & std::use_facet (
    const locale & __loc )
```

Return a facet.

use\_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has\_facet(locale) is true, there is a suitable facet to return. It throws std::bad\_cast if the locale doesn't contain a facet of type Facet.

## Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

**Parameters**

<code>__loc</code>	The locale to use.
--------------------	--------------------

**Returns**

Reference to facet of type `Facet`.

**Exceptions**

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
----------------------------	---

Definition at line 132 of file `locale_classes.tcc`.

## 3.47 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



## Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float std::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double std::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpa, typename _Tpb >`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta` (\_Tpa \_\_a, \_Tpb \_\_b)
- `float std::betaf` (float \_\_a, float \_\_b)
- `long double std::betal` (long double \_\_a, long double \_\_b)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1` (\_Tp \_\_k)
- `float std::comp_ellint_1f` (float \_\_k)
- `long double std::comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2` (\_Tp \_\_k)
- `float std::comp_ellint_2f` (float \_\_k)
- `long double std::comp_ellint_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- `float std::comp_ellint_3f` (float \_\_k, float \_\_nu)
- `long double std::comp_ellint_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::cyl_bessel_if` (float \_\_nu, float \_\_x)
- `long double std::cyl_bessel_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::cyl_bessel_jf` (float \_\_nu, float \_\_x)
- `long double std::cyl_bessel_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k` (\_Tpnu \_\_nu, \_Tp \_\_x)

- float `std::cyl_bessel_kf` (float \_\_nu, float \_\_x)
- long double `std::cyl_bessel_kl` (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2<\_Tpnu, \_Tp >::\_\_type `std::cyl_neumann` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::cyl_neumannf` (float \_\_nu, float \_\_x)
- long double `std::cyl_neumannl` (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp >::\_\_type `std::ellint_1` (\_Tp \_\_k, \_Tpp \_\_phi)
- float `std::ellint_1f` (float \_\_k, float \_\_phi)
- long double `std::ellint_1l` (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp >::\_\_type `std::ellint_2` (\_Tp \_\_k, \_Tpp \_\_phi)
- float `std::ellint_2f` (float \_\_k, float \_\_phi)
- long double `std::ellint_2l` (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3<\_Tp, \_Tpn, \_Tpp >::\_\_type `std::ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float `std::ellint_3f` (float \_\_k, float \_\_nu, float \_\_phi)
- long double `std::ellint_3l` (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::expint` (\_Tp \_\_x)
- float `std::expintf` (float \_\_x)
- long double `std::expintl` (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::hermite` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::hermitef` (unsigned int \_\_n, float \_\_x)
- long double `std::hermitel` (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::laguerre` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::laguerref` (unsigned int \_\_n, float \_\_x)
- long double `std::laguerrel` (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::legendre` (unsigned int \_\_l, \_Tp \_\_x)
- float `std::legendref` (unsigned int \_\_l, float \_\_x)
- long double `std::legendrel` (unsigned int \_\_l, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::riemann_zeta` (\_Tp \_\_s)
- float `std::riemann_zetaf` (float \_\_s)
- long double `std::riemann_zetal` (long double \_\_s)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::sph_bessel` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::sph_besself` (unsigned int \_\_n, float \_\_x)
- long double `std::sph_bessell` (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::sph_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float `std::sph_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double `std::sph_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type `std::sph_neumann` (unsigned int \_\_n, \_Tp \_\_x)
- float `std::sph_neumannf` (unsigned int \_\_n, float \_\_x)
- long double `std::sph_neumannl` (unsigned int \_\_n, long double \_\_x)

## 3.47.1 Detailed Description

A collection of advanced mathematical special functions, defined by ISO/IEC IS 29124.

## 3.47.2 Function Documentation

3.47.2.1 `assoc_laguerre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::assoc_laguerre (
    unsigned int __n,
    unsigned int __m,
    _Tp __x ) [inline]
```

Return the associated Laguerre polynomial of nonnegative order  $n$ , nonnegative degree  $m$  and real argument  $x \leftarrow$  :  $L_n^m(x)$ .

The associated Laguerre function of real degree  $\alpha$ ,  $L_n^\alpha(x)$ , is defined by

$$L_n^\alpha(x) = \frac{(\alpha+1)_n}{n!} {}_1F_1(-n; \alpha+1; x)$$

where  $(\alpha)_n$  is the Pochhammer symbol and  ${}_1F_1(a; c; x)$  is the confluent hypergeometric function.

The associated Laguerre polynomial is defined for integral degree  $\alpha = m$  by:

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x)$$

where the Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

and  $x \geq 0$ .

See also

`laguerre` for details of the Laguerre function of degree  $n$

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__n</code>	The order of the Laguerre function, <code>__n</code> $\geq 0$ .
------------------	---

**Parameters**

$\leftrightarrow$ __m	The degree of the Laguerre function, __m >= 0.
$\leftrightarrow$ __x	The argument of the Laguerre function, __x >= 0.

**Exceptions**

<code>std::domain_error</code>	if __x < 0.
--------------------------------	-------------

Definition at line 252 of file specfun.h.

**3.47.2.2 assoc\_laguerref()**

```
float std::assoc_laguerref (
    unsigned int __n,
    unsigned int __m,
    float __x ) [inline]
```

Return the associated Laguerre polynomial of order n, degree m:  $L_n^m(x)$  for float argument.

**See also**

assoc\_laguerre for more details.

Definition at line 206 of file specfun.h.

**3.47.2.3 assoc\_laguerrel()**

```
long double std::assoc_laguerrel (
    unsigned int __n,
    unsigned int __m,
    long double __x ) [inline]
```

Return the associated Laguerre polynomial of order n, degree m:  $L_n^m(x)$ .

**See also**

assoc\_laguerre for more details.

Definition at line 216 of file specfun.h.

3.47.2.4 `assoc_legendre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::assoc_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __x ) [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

The associated Legendre function is derived from the Legendre function  $P_l(x)$  by the Rodrigues formula:

$$P_l^m(x) = (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_l(x)$$

See also

`legendre` for details of the Legendre function of degree `l`

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__l</code>	The degree <code>__l</code> $\geq 0$ .
<code>__m</code>	The order <code>__m</code> $\leq l$ .
<code>__x</code>	The argument, <code>abs (__x) <math>\leq 1</math></code> .

## Exceptions

<code>std::domain_error</code>	if <code>abs (__x) &gt; 1</code> .
--------------------------------	------------------------------------

Definition at line 298 of file `specfun.h`.

3.47.2.5 `assoc_legendref()`

```
float std::assoc_legendref (
    unsigned int __l,
    unsigned int __m,
    float __x ) [inline]
```

Return the associated Legendre function of degree `l` and order `m` for `float` argument.



See also

`assoc_legendre` for more details.

Definition at line 267 of file `specfun.h`.

### 3.47.2.6 `assoc_legendrel()`

```
long double std::assoc_legendrel (
    unsigned int __l,
    unsigned int __m,
    long double __x ) [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

See also

`assoc_legendre` for more details.

Definition at line 276 of file `specfun.h`.

### 3.47.2.7 `beta()`

```
template<typename _Tpa , typename _Tpb >
__gnu_cxx::__promote_2<_Tpa, _Tpb>::__type std::beta (
    _Tpa __a,
    _Tpb __b ) [inline]
```

Return the beta function,  $B(a, b)$ , for real parameters `a`, `b`.

The beta function is defined by

$$B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1} dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

where  $a > 0$  and  $b > 0$

#### Template Parameters

<code>_Tpa</code>	The floating-point type of the parameter <code>__a</code> .
<code>_Tpb</code>	The floating-point type of the parameter <code>__b</code> .

#### Parameters

<code>__a</code>	The first argument of the beta function, $__a > 0$ .
------------------	--

## Parameters

<code>__b</code>	The second argument of the beta function, <code>__b &gt; 0</code> .
------------------	---

## Exceptions

<code>std::domain_error</code>	if <code>__a &lt; 0</code> or <code>__b &lt; 0</code> .
--------------------------------	---

Definition at line 343 of file `specfun.h`.

3.47.2.8 `betaf()`

```
float std::betaf (
    float __a,
    float __b ) [inline]
```

Return the beta function,  $B(a, b)$ , for `float` parameters `a`, `b`.

## See also

`beta` for more details.

Definition at line 312 of file `specfun.h`.

3.47.2.9 `betal()`

```
long double std::betal (
    long double __a,
    long double __b ) [inline]
```

Return the beta function,  $B(a, b)$ , for long double parameters `a`, `b`.

## See also

`beta` for more details.

Definition at line 322 of file `specfun.h`.

### 3.47.2.10 comp\_ellint\_1()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::comp_ellint_1 (
    _Tp __k ) [inline]
```

Return the complete elliptic integral of the first kind  $K(k)$  for real modulus  $k$ .

The complete elliptic integral of the first kind is defined as

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

where  $F(k, \phi)$  is the incomplete elliptic integral of the first kind and the modulus  $|k| \leq 1$ .

#### See also

ellint\_1 for details of the incomplete elliptic function of the first kind.

#### Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
------------------	---

#### Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
------------------	---

#### Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

Definition at line 391 of file specfun.h.

### 3.47.2.11 comp\_ellint\_1f()

```
float std::comp_ellint_1f (
    float __k ) [inline]
```

Return the complete elliptic integral of the first kind  $E(k)$  for `float` modulus  $k$ .

#### See also

comp\_ellint\_1 for details.

Definition at line 358 of file specfun.h.

3.47.2.12 `comp_ellint_1l()`

```
long double std::comp_ellint_1l (
    long double __k ) [inline]
```

Return the complete elliptic integral of the first kind  $E(k)$  for long double modulus  $k$ .

See also

`comp_ellint_1` for details.

Definition at line 368 of file `specfun.h`.

3.47.2.13 `comp_ellint_2()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::comp_ellint_2 (
    _Tp __k ) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for real modulus  $k$ .

The complete elliptic integral of the second kind is defined as

$$E(k) = E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta$$

where  $E(k, \phi)$  is the incomplete elliptic integral of the second kind and the modulus  $|k| \leq 1$ .

See also

`ellint_2` for details of the incomplete elliptic function of the second kind.

## Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
------------------	---

## Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
------------------	---

## Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

Definition at line 438 of file specfun.h.

#### 3.47.2.14 `comp_ellint_2f()`

```
float std::comp_ellint_2f (
    float __k ) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for `float` modulus `k`.

See also

`comp_ellint_2` for details.

Definition at line 406 of file specfun.h.

#### 3.47.2.15 `comp_ellint_2l()`

```
long double std::comp_ellint_2l (
    long double __k ) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for long double modulus `k`.

See also

`comp_ellint_2` for details.

Definition at line 416 of file specfun.h.

#### 3.47.2.16 `comp_ellint_3()`

```
template<typename _Tp , typename _Tpn >
__gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::comp_ellint_3 (
    _Tp __k,
    _Tpn __nu ) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu) = \Pi(k, \nu, \pi/2)$  for real modulus `k`.

The complete elliptic integral of the third kind is defined as

$$\Pi(k, \nu) = \Pi(k, \nu, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

where  $\Pi(k, \nu, \phi)$  is the incomplete elliptic integral of the second kind and the modulus  $|k| \leq 1$ .

See also

`ellint_3` for details of the incomplete elliptic function of the third kind.

## Template Parameters

<code>__Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>__Tpn</code>	The floating-point type of the argument <code>__nu</code> .

## Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__nu</code>	The argument

## Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

Definition at line 489 of file `specfun.h`.

3.47.2.17 `comp_ellint_3f()`

```
float std::comp_ellint_3f (
    float __k,
    float __nu ) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu)$  for `float` modulus `k`.

## See also

`comp_ellint_3` for details.

Definition at line 453 of file `specfun.h`.

3.47.2.18 `comp_ellint_3l()`

```
long double std::comp_ellint_3l (
    long double __k,
    long double __nu ) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu)$  for `long double` modulus `k`.

## See also

`comp_ellint_3` for details.

Definition at line 463 of file `specfun.h`.

### 3.47.2.19 cyl\_bessel\_i()

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_i (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for real order  $\nu$  and argument  $x \geq 0$ .

The regular modified cylindrical Bessel function is:

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

#### Template Parameters

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

#### Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

#### Exceptions

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

Definition at line 535 of file specfun.h.

### 3.47.2.20 cyl\_bessel\_if()

```
float std::cyl_bessel_if (
    float __nu,
    float __x ) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for `float` order  $\nu$  and argument  $x \geq 0$ .

#### See also

`cyl_bessel_i` for setails.

Definition at line 504 of file specfun.h.

3.47.2.21 `cyl_bessel_il()`

```
long double std::cyl_bessel_il (
    long double __nu,
    long double __x ) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for long double order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_i` for setails.

Definition at line 514 of file `specfun.h`.

3.47.2.22 `cyl_bessel_j()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_j (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the Bessel function  $J_\nu(x)$  of real order  $\nu$  and argument  $x \geq 0$ .

The cylindrical Bessel function is:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

## Template Parameters

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

## Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

## Exceptions

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

Definition at line 581 of file `specfun.h`.



### 3.47.2.23 cyl\_bessel\_jf()

```
float std::cyl_bessel_jf (
    float __nu,
    float __x ) [inline]
```

Return the Bessel function of the first kind  $J_\nu(x)$  for `float` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_j` for setails.

Definition at line 550 of file `specfun.h`.

### 3.47.2.24 cyl\_bessel\_jl()

```
long double std::cyl_bessel_jl (
    long double __nu,
    long double __x ) [inline]
```

Return the Bessel function of the first kind  $J_\nu(x)$  for `long double` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_j` for setails.

Definition at line 560 of file `specfun.h`.

### 3.47.2.25 cyl\_bessel\_k()

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_k (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  of real order  $\nu$  and argument  $x$ .

The irregular modified Bessel function is defined by:

$$K_\nu(x) = \frac{\pi}{2} \frac{I_{-\nu}(x) - I_\nu(x)}{\sin \nu\pi}$$

where for integral  $\nu = n$  a limit is taken:  $\lim_{\nu \rightarrow n}$ . For negative argument we have simply:

$$K_{-\nu}(x) = K_\nu(x)$$

## Template Parameters

<code>__Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>__Tp</code>	The floating-point type of the argument <code>__x</code> .

## Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x &gt;= 0</code>

## Exceptions

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------

Definition at line 633 of file `specfun.h`.

3.47.2.26 `cyl_bessel_kf()`

```
float std::cyl_bessel_kf (
    float __nu,
    float __x ) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  for `float` order  $\nu$  and argument  $x \geq 0$ .

## See also

`cyl_bessel_k` for details.

Definition at line 596 of file `specfun.h`.

3.47.2.27 `cyl_bessel_kl()`

```
long double std::cyl_bessel_kl (
    long double __nu,
    long double __x ) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  for `long double` order  $\nu$  and argument  $x \geq 0$ .

## See also

`cyl_bessel_k` for details.

Definition at line 606 of file `specfun.h`.

### 3.47.2.28 cyl\_neumann()

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_neumann (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the Neumann function  $N_\nu(x)$  of real order  $\nu$  and argument  $x \geq 0$ .

The Neumann function is defined by:

$$N_\nu(x) = \frac{J_\nu(x) \cos \nu\pi - J_{-\nu}(x)}{\sin \nu\pi}$$

where  $x \geq 0$  and for integral order  $\nu = n$  a limit is taken:  $\lim_{\nu \rightarrow n}$ .

#### Template Parameters

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

#### Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> $\geq 0$

#### Exceptions

<code>std::domain_error</code>	if <code>__x</code> $< 0$ .
--------------------------------	-----------------------------

Definition at line 681 of file specfun.h.

### 3.47.2.29 cyl\_neumannf()

```
float std::cyl_neumannf (
    float __nu,
    float __x ) [inline]
```

Return the Neumann function  $N_\nu(x)$  of `float` order  $\nu$  and argument  $x$ .

#### See also

`cyl_neumann` for setails.

Definition at line 648 of file specfun.h.

## 3.47.2.30 cyl\_neumannl()

```
long double std::cyl_neumannl (
    long double __nu,
    long double __x ) [inline]
```

Return the Neumann function  $N_\nu(x)$  of `long double` order  $\nu$  and argument  $x$ .

See also

`cyl_neumann` for setails.

Definition at line 658 of file `specfun.h`.

## 3.47.2.31 ellint\_1()

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::ellint_1 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

Return the incomplete elliptic integral of the first kind  $F(k, \phi)$  for `real` modulus  $k$  and angle  $\phi$ .

The incomplete elliptic integral of the first kind is defined as

$$F(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the first kind,  $K(k)$ .

See also

`comp_ellint_1`.

## Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

## Parameters

<code>__k</code>	The modulus, <code>abs(__k) &lt;= 1</code>
<code>__phi</code>	The integral limit argument in radians

**Exceptions**

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1 .</code>
--------------------------------	------------------------------------

Definition at line 729 of file `specfun.h`.

**3.47.2.32 ellint\_1f()**

```
float std::ellint_1f (
    float __k,
    float __phi ) [inline]
```

Return the incomplete elliptic integral of the first kind  $E(k, \phi)$  for `float` modulus  $k$  and angle  $\phi$ .

**See also**

`ellint_1` for details.

Definition at line 696 of file `specfun.h`.

**3.47.2.33 ellint\_1l()**

```
long double std::ellint_1l (
    long double __k,
    long double __phi ) [inline]
```

Return the incomplete elliptic integral of the first kind  $E(k, \phi)$  for `long double` modulus  $k$  and angle  $\phi$ .

**See also**

`ellint_1` for details.

Definition at line 706 of file `specfun.h`.

**3.47.2.34 ellint\_2()**

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::ellint_2 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$ .

The incomplete elliptic integral of the second kind is defined as

$$E(k, \phi) = \int_0^\phi \sqrt{1 - k^2 \sin^2 \theta}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the second kind,  $E(k)$ .

**See also**

`comp_ellint_2`.

## Template Parameters

<code>__Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>__Tpp</code>	The floating-point type of the angle <code>__phi</code> .

## Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__phi</code>	The integral limit argument in radians

## Returns

The elliptic function of the second kind.

## Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

Definition at line 777 of file `specfun.h`.

3.47.2.35 `ellint_2f()`

```
float std::ellint_2f (
    float __k,
    float __phi ) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$  for `float` argument.

## See also

`ellint_2` for details.

Definition at line 744 of file `specfun.h`.

3.47.2.36 `ellint_2l()`

```
long double std::ellint_2l (
    long double __k,
    long double __phi ) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$ .

## See also

`ellint_2` for details.

Definition at line 754 of file `specfun.h`.

### 3.47.2.37 `ellint_3()`

```
template<typename _Tp , typename _Tpn , typename _Tpp >
__gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::ellint_3 (
    _Tp __k,
    _Tpn __nu,
    _Tpp __phi ) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$ .

The incomplete elliptic integral of the third kind is defined by:

$$\Pi(k, \nu, \phi) = \int_0^\phi \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the third kind,  $\Pi(k, \nu)$ .

See also

`comp_ellint_3`.

#### Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpn</code>	The floating-point type of the argument <code>__nu</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

#### Parameters

<code>__k</code>	The modulus, <code>abs (__k) &lt;= 1</code>
<code>__nu</code>	The second argument
<code>__phi</code>	The integral limit argument in radians

#### Returns

The elliptic function of the third kind.

#### Exceptions

<code>std::domain_error</code>	if <code>abs (__k) &gt; 1</code> .
--------------------------------	------------------------------------

Definition at line 830 of file `specfun.h`.

### 3.47.2.38 `ellint_3f()`

```
float std::ellint_3f (
    float __k,
```

```
float __nu,
float __phi ) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$  for `float` argument.

See also

`ellint_3` for details.

Definition at line 792 of file `specfun.h`.

### 3.47.2.39 `ellint_3l()`

```
long double std::ellint_3l (
    long double __k,
    long double __nu,
    long double __phi ) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$ .

See also

`ellint_3` for details.

Definition at line 802 of file `specfun.h`.

### 3.47.2.40 `expint()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::expint (
    _Tp __x ) [inline]
```

Return the exponential integral  $Ei(x)$  for `real` argument `x`.

The exponential integral is given by

$$Ei(x) = - \int_{-x}^{\infty} \frac{e^t}{t} dt$$

#### Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--



**Parameters**

<code>_↔ _x</code>	The argument of the exponential integral function.
------------------------	--

Definition at line 870 of file specfun.h.

**3.47.2.41 expintf()**

```
float std::expintf (
    float __x ) [inline]
```

Return the exponential integral  $Ei(x)$  for `float` argument `x`.

**See also**

`expint` for details.

Definition at line 844 of file specfun.h.

**3.47.2.42 expintl()**

```
long double std::expintl (
    long double __x ) [inline]
```

Return the exponential integral  $Ei(x)$  for `long double` argument `x`.

**See also**

`expint` for details.

Definition at line 854 of file specfun.h.

**3.47.2.43 hermite()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::hermite (
    unsigned int __n,
    _Tp __x ) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of order `n` and `real` argument `x`.

The Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$$

The Hermite polynomial obeys a reflection formula:

$$H_n(-x) = (-1)^n H_n(x)$$

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__n</code>	The order
<code>__x</code>	The argument

Definition at line 918 of file `specfun.h`.

3.47.2.44 `hermitef()`

```
float std::hermitef (
    unsigned int __n,
    float __x ) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of nonnegative order `n` and float argument `x`.

## See also

`hermite` for details.

Definition at line 885 of file `specfun.h`.

3.47.2.45 `hermitel()`

```
long double std::hermitel (
    unsigned int __n,
    long double __x ) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of nonnegative order `n` and long double argument `x`.

## See also

`hermite` for details.

Definition at line 895 of file `specfun.h`.

3.47.2.46 `laguerre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::laguerre (
    unsigned int __n,
    _Tp __x ) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree `n` and real argument  $x \geq 0$ .

The Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

**Template Parameters**

<code>__Tp</code>	The floating-point type of the argument <code>__x</code> .
-------------------	--

**Parameters**

<code>__n</code>	The nonnegative order
<code>__x</code>	The argument <code>__x &gt;= 0</code>

**Exceptions**

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------

Definition at line 962 of file `specfun.h`.

**3.47.2.47 `laguerref()`**

```
float std::laguerref (
    unsigned int __n,
    float __x ) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree `n` and `float` argument  $x \geq 0$ .

**See also**

`laguerre` for more details.

Definition at line 933 of file `specfun.h`.

**3.47.2.48 `laguerrel()`**

```
long double std::laguerrel (
    unsigned int __n,
    long double __x ) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree `n` and `long double` argument  $x \geq 0$ .

**See also**

`laguerre` for more details.

Definition at line 943 of file `specfun.h`.

3.47.2.49 `legendre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::legendre (
    unsigned int __l,
    _Tp __x ) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and real argument  $|x| \leq 0$ .

The Legendre function of order  $l$  and argument  $x$ ,  $P_l(x)$ , is defined by:

$$P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} (x^2 - 1)^l$$

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__l</code>	The degree $l \geq 0$
<code>__x</code>	The argument $\text{abs}(\text{__x}) \leq 1$

## Exceptions

<code>std::domain_error</code>	if $\text{abs}(\text{__x}) > 1$
--------------------------------	---------------------------------

Definition at line 1007 of file `specfun.h`.

3.47.2.50 `legendref()`

```
float std::legendref (
    unsigned int __l,
    float __x ) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and `float` argument  $|x| \leq 0$ .

## See also

`legendre` for more details.

Definition at line 977 of file `specfun.h`.

### 3.47.2.51 legendrel()

```
long double std::legendrel (
    unsigned int __l,
    long double __x ) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and long double argument  $|x| \leq 0$ .

See also

legendre for more details.

Definition at line 987 of file specfun.h.

### 3.47.2.52 riemann\_zeta()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::riemann_zeta (
    _Tp __s ) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for real argument  $s$ .

The Riemann zeta function is defined by:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \text{ for } s > 1$$

and

$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{k=1}^{\infty} (-1)^{k-1} k^{-s} \text{ for } 0 \leq s \leq 1$$

For  $s < 1$  use the reflection formula:

$$\zeta(s) = 2^s \pi^{s-1} \sin\left(\frac{\pi s}{2}\right) \Gamma(1-s) \zeta(1-s)$$

#### Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__s</code> .
------------------	--

#### Parameters

<code>__s</code>	The argument $s \neq 1$
------------------	-------------------------

Definition at line 1058 of file specfun.h.

3.47.2.53 `riemann_zetaf()`

```
float std::riemann_zetaf (
    float __s ) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for `float` argument  $s$ .

See also

`riemann_zeta` for more details.

Definition at line 1022 of file `specfun.h`.

3.47.2.54 `riemann_zetal()`

```
long double std::riemann_zetal (
    long double __s ) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for `long double` argument  $s$ .

See also

`riemann_zeta` for more details.

Definition at line 1032 of file `specfun.h`.

3.47.2.55 `sph_bessel()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::sph_bessel (
    unsigned int __n,
    _Tp __x ) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order  $n$  and real argument  $x \geq 0$ .

The spherical Bessel function is defined by:

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} J_{n+1/2}(x)$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

**Parameters**

$\leftrightarrow$ __n	The integral order $n \geq 0$
$\leftrightarrow$ __x	The real argument $x \geq 0$

**Exceptions**

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------

Definition at line 1102 of file `specfun.h`.

**3.47.2.56 sph\_besself()**

```
float std::sph_besself (
    unsigned int __n,
    float __x ) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order  $n$  and `float` argument  $x \geq 0$ .

**See also**

`sph_bessel` for more details.

Definition at line 1073 of file `specfun.h`.

**3.47.2.57 sph\_bessell()**

```
long double std::sph_bessell (
    unsigned int __n,
    long double __x ) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order  $n$  and `long double` argument  $x \geq 0$ .

**See also**

`sph_bessel` for more details.

Definition at line 1083 of file `specfun.h`.

## 3.47.2.58 sph\_legendre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __theta ) [inline]
```

Return the spherical Legendre function of nonnegative integral degree  $l$  and order  $m$  and real angle  $\theta$  in radians.

The spherical Legendre function is defined by

$$Y_l^m(\theta, \phi) = (-1)^m \left[ \frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!} \right] P_l^m(\cos \theta) \exp^{im\phi}$$

## Template Parameters

<code>_Tp</code>	The floating-point type of the angle <code>__theta</code> .
------------------	---

## Parameters

<code>__l</code>	The order <code>__l</code> $\geq 0$
<code>__m</code>	The degree <code>__m</code> $\geq 0$ and <code>__m</code> $\leq$ <code>__l</code>
<code>__theta</code>	The radian polar angle argument

Definition at line 1149 of file specfun.h.

## 3.47.2.59 sph\_legendref()

```
float std::sph_legendref (
    unsigned int __l,
    unsigned int __m,
    float __theta ) [inline]
```

Return the spherical Legendre function of nonnegative integral degree  $l$  and order  $m$  and float angle  $\theta$  in radians.

## See also

`sph_legendre` for details.

Definition at line 1117 of file specfun.h.



## 3.47.2.60 sph\_legendrel()

```
long double std::sph_legendrel (
    unsigned int __l,
    unsigned int __m,
    long double __theta ) [inline]
```

Return the spherical Legendre function of nonnegative integral degree  $l$  and order  $m$  and long double angle  $\theta$  in radians.

See also

sph\_legendre for details.

Definition at line 1128 of file specfun.h.

## 3.47.2.61 sph\_neumann()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::sph_neumann (
    unsigned int __n,
    _Tp __x ) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and real argument  $x \geq 0$ .

The spherical Neumann function is defined by

$$n_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} N_{n+1/2}(x)$$

## Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

## Parameters

<code>__n</code>	The integral order $n \geq 0$
<code>__x</code>	The real argument $__x \geq 0$

## Exceptions

<code>std::domain_error</code>	if <code>__x &lt; 0</code> .
--------------------------------	------------------------------

Definition at line 1193 of file specfun.h.

**3.47.2.62 sph\_neumannf()**

```
float std::sph_neumannf (
    unsigned int __n,
    float __x ) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and `float` argument  $x \geq 0$ .

See also

`sph_neumann` for details.

Definition at line 1164 of file `specfun.h`.

**3.47.2.63 sph\_neumannl()**

```
long double std::sph_neumannl (
    unsigned int __n,
    long double __x ) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and `long double`  $x \geq 0$ .

See also

`sph_neumann` for details.

Definition at line 1174 of file `specfun.h`.

### 3.48 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



#### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float std::tr1::betaf` (float \_\_x, float \_\_y)
- `long double std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_1f` (float \_\_k)
- `long double std::tr1::comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_2f` (float \_\_k)
- `long double std::tr1::comp_ellint_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- `float std::tr1::comp_ellint_3f` (float \_\_k, float \_\_nu)
- `long double std::tr1::comp_ellint_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_if` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_jf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k` (\_Tpnu \_\_nu, \_Tp \_\_x)

- float **std::tr1::cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2<\_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3<\_Tp, \_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote<\_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

### 3.48.1 Detailed Description

A collection of advanced mathematical special functions.

### 3.48.2 Function Documentation

#### 3.48.2.1 `assoc_laguerre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_laguerre (
    unsigned int __n,
    unsigned int __m,
    _Tp __x ) [inline]
```

#### 5.2.1.1 Associated Laguerre polynomials.

Definition at line 1274 of file tr1/cmath.

#### 3.48.2.2 `assoc_legendre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __x ) [inline]
```

#### 5.2.1.2 Associated Legendre functions.

Definition at line 1291 of file tr1/cmath.

#### 3.48.2.3 `beta()`

```
template<typename _Tpx , typename _Tpy >
__gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta (
    _Tpx __x,
    _Tpy __y ) [inline]
```

#### 5.2.1.3 Beta functions.

Definition at line 1308 of file tr1/cmath.

#### 3.48.2.4 comp\_ellint\_1()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_1 (
    _Tp __k ) [inline]
```

##### 5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 1325 of file tr1/cmath.

#### 3.48.2.5 comp\_ellint\_2()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_2 (
    _Tp __k ) [inline]
```

##### 5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 1342 of file tr1/cmath.

#### 3.48.2.6 comp\_ellint\_3()

```
template<typename _Tp , typename _Tpn >
__gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3 (
    _Tp __k,
    _Tpn __nu ) [inline]
```

##### 5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 1359 of file tr1/cmath.

#### 3.48.2.7 cyl\_bessel\_i()

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

##### 5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 1376 of file tr1/cmath.

### 3.48.2.8 `cyl_bessel_j()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

#### 5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 1393 of file tr1/cmath.

### 3.48.2.9 `cyl_bessel_k()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_k (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

#### 5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 1410 of file tr1/cmath.

### 3.48.2.10 `cyl_neumann()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_neumann (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

#### 5.2.1.11 Cylindrical Neumann functions.

Definition at line 1427 of file tr1/cmath.

### 3.48.2.11 `ellint_1()`

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

#### 5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 1444 of file tr1/cmath.

**3.48.2.12 ellint\_2()**

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 1461 of file tr1/cmath.

**3.48.2.13 ellint\_3()**

```
template<typename _Tp , typename _Tpn , typename _Tpp >
__gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::tr1::ellint_3 (
    _Tp __k,
    _Tpn __nu,
    _Tpp __phi ) [inline]
```

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 1478 of file tr1/cmath.

**3.48.2.14 expint()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::expint (
    _Tp __x ) [inline]
```

5.2.1.15 Exponential integrals.

Definition at line 1495 of file tr1/cmath.

**3.48.2.15 hermite()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::hermite (
    unsigned int __n,
    _Tp __x ) [inline]
```

5.2.1.16 Hermite polynomials.

Definition at line 1512 of file tr1/cmath.



**3.48.2.16 laguerre()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::laguerre (
    unsigned int __n,
    _Tp __x ) [inline]
```

**5.2.1.18 Laguerre polynomials.**

Definition at line 1529 of file tr1/cmath.

**3.48.2.17 legendre()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::legendre (
    unsigned int __n,
    _Tp __x ) [inline]
```

**5.2.1.19 Legendre polynomials.**

Definition at line 1546 of file tr1/cmath.

**3.48.2.18 riemann\_zeta()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::riemann_zeta (
    _Tp __x ) [inline]
```

**5.2.1.20 Riemann zeta function.**

Definition at line 1563 of file tr1/cmath.

**3.48.2.19 sph\_bessel()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_bessel (
    unsigned int __n,
    _Tp __x ) [inline]
```

**5.2.1.21 Spherical Bessel functions.**

Definition at line 1580 of file tr1/cmath.

## 3.48.2.20 sph\_legendre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __theta ) [inline]
```

## 5.2.1.22 Spherical associated Legendre functions.

Definition at line 1597 of file tr1/cmath.

## 3.48.2.21 sph\_neumann()

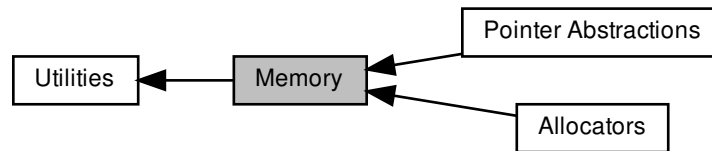
```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_neumann (
    unsigned int __n,
    _Tp __x ) [inline]
```

## 5.2.1.23 Spherical Neumann functions.

Definition at line 1614 of file tr1/cmath.

### 3.49 Memory

Collaboration diagram for Memory:



#### Modules

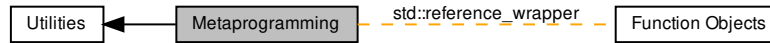
- [Allocators](#)
- [Pointer Abstractions](#)

#### 3.49.1 Detailed Description

Components for memory allocation, deallocation, and management.

## 3.50 Metaprogramming

Collaboration diagram for Metaprogramming:



## Classes

- struct `std::__add_pointer_helper< _Tp, bool >`
- struct `std::__detector< _Default, _AlwaysVoid, _Op, _Args >`
- struct `std::__detector< _Default, __void_t< _Op< _Args... > >, _Op, _Args... >`
- struct `std::__is_nullptr_t< _Tp >`
- struct `std::__is_trivially_copy_assignable_impl< _Tp, bool >`
- struct `std::__is_trivially_copy_constructible_impl< _Tp, bool >`
- struct `std::__is_trivially_move_assignable_impl< _Tp, bool >`
- struct `std::__is_trivially_move_constructible_impl< _Tp, bool >`
- struct `std::add_const< _Tp >`
- struct `std::add_cv< _Tp >`
- struct `std::add_lvalue_reference< _Tp >`
- struct `std::add_rvalue_reference< _Tp >`
- struct `std::add_volatile< _Tp >`
- struct `std::aligned_storage< _Len, _Align >`
- struct `std::aligned_union< _Len, _Types >`
- struct `std::alignment_of< _Tp >`
- struct `std::common_type< _Tp >`
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- class `std::decay< _Tp >`
- struct `std::enable_if< bool, _Tp >`
- struct `std::extent< typename, _UInt >`
- struct `std::has_virtual_destructor< _Tp >`
- struct `std::integral_constant< _Tp, __v >`
- struct `std::is_abstract< _Tp >`
- struct `std::is_arithmetic< _Tp >`
- struct `std::is_array< typename >`
- struct `std::is_assignable< _Tp, _Up >`
- struct `std::is_base_of< _Base, _Derived >`
- struct `std::is_class< _Tp >`
- struct `std::is_compound< _Tp >`
- struct `std::is_const< typename >`
- struct `std::is_constructible< _Tp, _Args >`
- struct `std::is_convertible< _From, _To >`
- struct `std::is_copy_assignable< _Tp >`
- struct `std::is_copy_constructible< _Tp >`
- struct `std::is_default_constructible< _Tp >`

- struct `std::is_destructible< _Tp >`
- struct `std::is_empty< _Tp >`
- struct `std::is_enum< _Tp >`
- struct `std::is_final< _Tp >`
- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< typename >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_null_pointer< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< typename, typename >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< typename >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< typename >`
- class `std::reference_wrapper< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`
- struct `std::remove_cv< _Tp >`
- struct `std::remove_extent< _Tp >`
- struct `std::remove_pointer< _Tp >`
- struct `std::remove_reference< _Tp >`

- struct `std::remove_volatile< _Tp >`
- class `std::result_of< _Signature >`
- struct `std::tr2::__reflection_typelist< _Elements >`
- struct `std::tr2::__reflection_typelist< _First, _Rest... >`
- struct `std::tr2::__reflection_typelist<>`
- struct `std::tr2::bases< _Tp >`
- struct `std::tr2::direct_bases< _Tp >`
- struct `std::underlying_type< _Tp >`

#### Macros

- `#define __cpp_lib_is_final`
- `#define __cpp_lib_is_null_pointer`
- `#define __cpp_lib_result_of_sfinae`
- `#define __cpp_lib_transformation_trait_aliases`
- `#define __cpp_lib_void_t`

#### Typedefs

- template<bool \_\_v>  
using `std::__bool_constant` = `integral_constant< bool, __v >`
- template<typename \_Default, template< typename... > class \_Op, typename... \_Args>  
using `std::__detected_or` = `__detector< _Default, void, _Op, _Args... >`
- template<typename \_Default, template< typename... > class \_Op, typename... \_Args>  
using `std::__detected_or_t` = `typename __detected_or< _Default, _Op, _Args... >::type`
- template<bool \_Cond, typename \_Tp = void>  
using `std::__enable_if_t` = `typename enable_if< _Cond, _Tp >::type`
- template<typename \_Tp, typename... \_Args>  
using `std::__is_nothrow_constructible_impl` = `__is_nt_constructible_impl< __is_constructible(_Tp, _Args...), _Tp, _Args... >`
- template<typename... >  
using `std::__void_t` = `void`
- template<typename... \_Cond>  
using `std::__Require` = `typename enable_if< __and< _Cond... >::value >::type`
- template<typename \_Tp >  
using `std::add_const_t` = `typename add_const< _Tp >::type`
- template<typename \_Tp >  
using `std::add_cv_t` = `typename add_cv< _Tp >::type`
- template<typename \_Tp >  
using `std::add_lvalue_reference_t` = `typename add_lvalue_reference< _Tp >::type`
- template<typename \_Tp >  
using `std::add_pointer_t` = `typename add_pointer< _Tp >::type`
- template<typename \_Tp >  
using `std::add_rvalue_reference_t` = `typename add_rvalue_reference< _Tp >::type`
- template<typename \_Tp >  
using `std::add_volatile_t` = `typename add_volatile< _Tp >::type`
- template<size\_t \_Len, size\_t \_Align = \_\_alignof\_\_(typename \_\_aligned\_storage\_msa< \_Len >::\_\_type)>  
using `std::aligned_storage_t` = `typename aligned_storage< _Len, _Align >::type`
- template<size\_t \_Len, typename... \_Types>  
using `std::aligned_union_t` = `typename aligned_union< _Len, _Types... >::type`

- `template<typename... _Tp>`  
    `using std::common_type_t = typename common_type< _Tp... >::type`
- `template<bool _Cond, typename _Iftrue, typename _Iffalse >`  
    `using std::conditional_t = typename conditional< _Cond, _Iftrue, _Iffalse >::type`
- `template<typename _Tp >`  
    `using std::decay_t = typename decay< _Tp >::type`
- `template<bool _Cond, typename _Tp = void>`  
    `using std::enable_if_t = typename enable_if< _Cond, _Tp >::type`
- `typedef integral_constant< bool, false > std::false_type`
- `template<typename _Tp >`  
    `using std::make_signed_t = typename make_signed< _Tp >::type`
- `template<typename _Tp >`  
    `using std::make_unsigned_t = typename make_unsigned< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_all_extents_t = typename remove_all_extents< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_const_t = typename remove_const< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_cv_t = typename remove_cv< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_extent_t = typename remove_extent< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_pointer_t = typename remove_pointer< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_reference_t = typename remove_reference< _Tp >::type`
- `template<typename _Tp >`  
    `using std::remove_volatile_t = typename remove_volatile< _Tp >::type`
- `template<typename _Tp >`  
    `using std::result_of_t = typename result_of< _Tp >::type`
- `typedef integral_constant< bool, true > std::true_type`
- `template<typename _Tp >`  
    `using std::underlying_type_t = typename underlying_type< _Tp >::type`
- `template<typename... >`  
    `using std::void_t = void`

## Functions

- `template<typename _Tp >`  
    `auto std::declval () noexcept -> decltype(__declval< _Tp >(0))`

## Variables

- `static const size_t std::aligned_union< _Len, _Types >::alignment_value`
- `static constexpr _Tp std::integral_constant< _Tp, __v >::value`

### 3.50.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

### 3.50.2 Typedef Documentation

#### 3.50.2.1 add\_const\_t

```
template<typename _Tp >  
using std::add_const_t = typedef typename add_const<_Tp>::type
```

Alias template for add\_const.

Definition at line 1389 of file type\_traits.

#### 3.50.2.2 add\_cv\_t

```
template<typename _Tp >  
using std::add_cv_t = typedef typename add_cv<_Tp>::type
```

Alias template for add\_cv.

Definition at line 1397 of file type\_traits.

#### 3.50.2.3 add\_lvalue\_reference\_t

```
template<typename _Tp >  
using std::add_lvalue_reference_t = typedef typename add_lvalue_reference<_Tp>::type
```

Alias template for add\_lvalue\_reference.

Definition at line 1450 of file type\_traits.

#### 3.50.2.4 add\_pointer\_t

```
template<typename _Tp >  
using std::add_pointer_t = typedef typename add_pointer<_Tp>::type
```

Alias template for add\_pointer.

Definition at line 1768 of file type\_traits.



#### 3.50.2.5 add\_rvalue\_reference\_t

```
template<typename _Tp >
using std::add_rvalue_reference_t = typedef typename add_rvalue_reference<_Tp>::type
```

Alias template for add\_rvalue\_reference.

Definition at line 1454 of file type\_traits.

#### 3.50.2.6 add\_volatile\_t

```
template<typename _Tp >
using std::add_volatile_t = typedef typename add_volatile<_Tp>::type
```

Alias template for add\_volatile.

Definition at line 1393 of file type\_traits.

#### 3.50.2.7 aligned\_storage\_t

```
template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>
using std::aligned_storage_t = typedef typename aligned_storage<_Len, _Align>::type
```

Alias template for aligned\_storage.

Definition at line 2276 of file type\_traits.

#### 3.50.2.8 common\_type\_t

```
template<typename... _Tp>
using std::common_type_t = typedef typename common_type<_Tp...>::type
```

Alias template for common\_type.

Definition at line 2295 of file type\_traits.

#### 3.50.2.9 conditional\_t

```
template<bool _Cond, typename _Iftrue , typename _Iffalse >
using std::conditional_t = typedef typename conditional<_Cond, _Iftrue, _Iffalse>::type
```

Alias template for conditional.

Definition at line 2291 of file type\_traits.

#### 3.50.2.10 decay\_t

```
template<typename _Tp >  
using std::decay_t = typedef typename decay<_Tp>::type
```

Alias template for decay.

Definition at line 2283 of file type\_traits.

#### 3.50.2.11 enable\_if\_t

```
template<bool _Cond, typename _Tp = void>  
using std::enable_if_t = typedef typename enable_if<_Cond, _Tp>::type
```

Alias template for enable\_if.

Definition at line 2287 of file type\_traits.

#### 3.50.2.12 false\_type

```
typedef integral_constant<bool, false> std::false_type
```

The type used as a compile-time boolean with false value.

Definition at line 78 of file type\_traits.

#### 3.50.2.13 make\_signed\_t

```
template<typename _Tp >  
using std::make_signed_t = typedef typename make_signed<_Tp>::type
```

Alias template for make\_signed.

Definition at line 1685 of file type\_traits.

#### 3.50.2.14 make\_unsigned\_t

```
template<typename _Tp >  
using std::make_unsigned_t = typedef typename make_unsigned<_Tp>::type
```

Alias template for make\_unsigned.

Definition at line 1689 of file type\_traits.

#### 3.50.2.15 `remove_all_extents_t`

```
template<typename _Tp >
using std::remove_all_extents_t = typedef typename remove_all_extents<_Tp>::type
```

Alias template for `remove_all_extents`.

Definition at line 1727 of file `type_traits`.

#### 3.50.2.16 `remove_const_t`

```
template<typename _Tp >
using std::remove_const_t = typedef typename remove_const<_Tp>::type
```

Alias template for `remove_const`.

Definition at line 1377 of file `type_traits`.

#### 3.50.2.17 `remove_cv_t`

```
template<typename _Tp >
using std::remove_cv_t = typedef typename remove_cv<_Tp>::type
```

Alias template for `remove_cv`.

Definition at line 1385 of file `type_traits`.

#### 3.50.2.18 `remove_extent_t`

```
template<typename _Tp >
using std::remove_extent_t = typedef typename remove_extent<_Tp>::type
```

Alias template for `remove_extent`.

Definition at line 1723 of file `type_traits`.

#### 3.50.2.19 `remove_pointer_t`

```
template<typename _Tp >
using std::remove_pointer_t = typedef typename remove_pointer<_Tp>::type
```

Alias template for `remove_pointer`.

Definition at line 1764 of file `type_traits`.

#### 3.50.2.20 remove\_reference\_t

```
template<typename _Tp >  
using std::remove_reference_t = typedef typename remove_reference<_Tp>::type
```

Alias template for remove\_reference.

Definition at line 1446 of file type\_traits.

#### 3.50.2.21 remove\_volatile\_t

```
template<typename _Tp >  
using std::remove_volatile_t = typedef typename remove_volatile<_Tp>::type
```

Alias template for remove\_volatile.

Definition at line 1381 of file type\_traits.

#### 3.50.2.22 result\_of\_t

```
template<typename _Tp >  
using std::result_of_t = typedef typename result_of<_Tp>::type
```

Alias template for result\_of.

Definition at line 2303 of file type\_traits.

#### 3.50.2.23 true\_type

```
typedef integral_constant<bool, true> std::true_type
```

The type used as a compile-time boolean with true value.

Definition at line 75 of file type\_traits.

#### 3.50.2.24 underlying\_type\_t

```
template<typename _Tp >  
using std::underlying_type_t = typedef typename underlying_type<_Tp>::type
```

Alias template for underlying\_type.

Definition at line 2299 of file type\_traits.

#### 3.50.2.25 void\_t

```
template<typename... >  
using std::void_t = typedef void
```

A metafunction that always yields void, used for detecting valid types.

Definition at line 2316 of file type\_traits.

### 3.50.3 Variable Documentation

#### 3.50.3.1 alignment\_value

```
template<size_t _Len, typename... _Types>  
const size_t std::aligned_union< _Len, _Types >::alignment_value [static]
```

The value of the strictest alignment of \_Types.

Definition at line 1841 of file type\_traits.

## 3.51 Mutating

Collaboration diagram for Mutating:



## Functions

- `template<typename _II, typename _OI >`  
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, __out_true, __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`

- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵  
Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_↵  
__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵  
Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_↵  
value)`
- `template<typename _BidirectionalIterator >`  
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator ↵  
__result)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::V2::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, ↵  
__OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber_↵  
Generator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↵  
__first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Unary_↵  
Operation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Output_↵  
Iterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Binary_↵  
Predicate __binary_pred)`

## 3.51.1 Detailed Description

## 3.51.2 Function Documentation

3.51.2.1 `copy()`

```
template<typename _II , typename _OI >
_OI std::copy (
    _II __first,
    _II __last,
    _OI __result ) [inline]
```

Copies the range `[first,last)` into `result`.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 446 of file `stl_algobase.h`.

3.51.2.2 `copy_backward()`

```
template<typename _BI1 , typename _BI2 >
_BI2 std::copy_backward (
    _BI1 __first,
    _BI1 __last,
    _BI2 __result ) [inline]
```

Copies the range `[first,last)` into `result`.

## Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.



**Returns**

result - (first - last)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `copy` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 622 of file `stl_algobase.h`.

**3.51.2.3 `copy_if()`**

```
template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
_OutputIterator std::copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred )
```

Copy the elements of a sequence for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `__result`. `copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 737 of file `stl_algo.h`.

**3.51.2.4 `copy_n()`**

```
template<typename _InputIterator , typename _Size , typename _OutputIterator >
_OutputIterator std::copy_n (
    _InputIterator __first,
    _Size __n,
    _OutputIterator __result ) [inline]
```

Copies the range `[first,first+n)` into `[result,result+n)`.

## Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

## Returns

`result+n`.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 799 of file `stl_algo.h`.

3.51.2.5 `fill()`

```
template<typename _ForwardIterator , typename _Tp >
void std::fill (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __value ) [inline]
```

Fills the range `[first,last)` with copies of `value`.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__value</code>	A reference-to-const of arbitrary type.

## Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 724 of file `stl_algobase.h`.

### 3.51.2.6 fill\_n()

```
template<typename _OI , typename _Size , typename _Tp >
_OI std::fill_n (
    _OI __first,
    _Size __n,
    const _Tp & __value ) [inline]
```

Fills the range [first,first+n) with copies of value.

## Parameters

<code>__first</code>	An output iterator.
<code>__n</code>	The count of copies to perform.
<code>__value</code>	A reference-to-const of arbitrary type.

## Returns

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 784 of file `stl_algo.h`.

3.51.2.7 `generate()`

```
template<typename _ForwardIterator, typename _Generator>
void std::generate (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Generator __gen )
```

Assign the result of a function object to each value in a sequence.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits&lt;_ForwardIterator&gt;::value_type</code>

## Returns

`generate()` returns no value.

Performs the assignment `*i = __gen()` for each `i` in the range `[__first,__last)`.

Definition at line 4426 of file `stl_algo.h`.

### 3.51.2.8 generate\_n()

```
template<typename _OutputIterator , typename _Size , typename _Generator >
_OutputIterator std::generate_n (
    _OutputIterator __first,
    _Size __n,
    _Generator __gen )
```

Assign the result of a function object to each value in a sequence.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__n</code>	The length of the sequence.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits&lt;_ForwardIterator&gt;::value_type</code>

#### Returns

The end of the sequence, `__first+__n`

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __first+__n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 4457 of file `stl_algo.h`.

### 3.51.2.9 is\_partitioned()

```
template<typename _InputIterator , typename _Predicate >
bool std::is_partitioned (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks whether the sequence is partitioned.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

#### Returns

True if the range `[__first, __last)` is partitioned by `__pred`, i.e. if all elements that satisfy `__pred` appear before those that do not.

Definition at line 582 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

#### 3.51.2.10 `iter_swap()`

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
void std::iter_swap (
    _ForwardIterator1 __a,
    _ForwardIterator2 __b ) [inline]
```

Swaps the contents of two iterators.

##### Parameters

<code>↔ __a</code>	An iterator.
<code>↔ __b</code>	Another iterator.

##### Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 120 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_to_first()`, `std::__partition()`, `std::__reverse()`, `std::__V2::__rotate()`, and `std::__unguarded_partition()`.

#### 3.51.2.11 `move()`

```
template<typename _II , typename _OI >
_OI std::move (
    _II __first,
    _II __last,
    _OI __result ) [inline]
```

Moves the range `[first,last)` into `result`.

##### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

**Returns**

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 479 of file `stl_algobase.h`.

**3.51.2.12 `move_backward()`**

```
template<typename _BI1 , typename _BI2 >
_BI2 std::move_backward (
    _BI1 __first,
    _BI1 __last,
    _BI2 __result ) [inline]
```

Moves the range `[first,last)` into `result`.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

**Returns**

result - (first - last)

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `move` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 658 of file `stl_algobase.h`.

**3.51.2.13 `partition()`**

```
template<typename _ForwardIterator , typename _Predicate >
_FowardIterator std::partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred ) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

## Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[__first,middle)` and false for each `i` in the range `[middle,__last)`.

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 4641 of file `stl_algo.h`.

3.51.2.14 `partition_copy()`

```
template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename
_Predicate >
pair<_OutputIterator1, _OutputIterator2> std::partition_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator1 __out_true,
    _OutputIterator2 __out_false,
    _Predicate __pred )
```

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__out_true</code>	An output iterator.
<code>__out_false</code>	An output iterator.
<code>__pred</code>	A predicate.

## Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `out_true` and each element for which `__pred` returns false to `__out_false`.

Definition at line 828 of file `stl_algo.h`.



### 3.51.2.15 partition\_point()

```
template<typename _ForwardIterator , typename _Predicate >
_FForwardIterator std::partition_point (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred )
```

Find the partition point of a partitioned range.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__pred</code>	A predicate.

#### Returns

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

Definition at line 603 of file `stl_algo.h`.

### 3.51.2.16 random\_shuffle()

```
template<typename _RandomAccessIterator , typename _RandomNumberGenerator >
void std::random_shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _RandomNumberGenerator && __rand )
```

Shuffle the elements of a sequence using a random number generator.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__rand</code>	The RNG functor or function.

#### Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__rand` to provide a random distribution. Calling `__rand(N)` for a positive integer `N` should return a randomly chosen integer from the range `[0,N)`.

Definition at line 4601 of file `stl_algo.h`.

3.51.2.17 `remove()`

```
template<typename _ForwardIterator , typename _Tp >
_FowardIterator std::remove (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __value ) [inline]
```

Remove elements from a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be removed.

## Returns

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range `[__first,__last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 896 of file `stl_algo.h`.

3.51.2.18 `remove_copy()`

```
template<typename _InputIterator , typename _OutputIterator , typename _Tp >
_OutputIterator std::remove_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    const _Tp & __value ) [inline]
```

Copy a sequence, removing elements of a given value.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__value</code>	The value to be removed.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` not equal to `__value` to the range beginning at `__result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 670 of file `stl_algo.h`.

**3.51.2.19 `remove_copy_if()`**

```
template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
_OutputIterator std::remove_copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred ) [inline]
```

Copy a sequence, removing elements for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

**Returns**

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 703 of file `stl_algo.h`.

**3.51.2.20 `remove_if()`**

```
template<typename _ForwardIterator , typename _Predicate >
_FowardIterator std::remove_if (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred ) [inline]
```

Remove elements from a sequence using a predicate.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.

## Returns

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range `[__first,__last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 929 of file `stl_algo.h`.

3.51.2.21 `replace()`

```
template<typename _ForwardIterator , typename _Tp >
void std::replace (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __old_value,
    const _Tp & __new_value )
```

Replace each occurrence of one value in a sequence with another value.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

## Returns

`replace()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `*i == __old_value` then the assignment `*i = __new_value` is performed.

Definition at line 4362 of file `stl_algo.h`.

### 3.51.2.22 replace\_copy\_if()

```
template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp
>
_OutputIterator std::replace_copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred,
    const _Tp & __new_value ) [inline]
```

Copy a sequence, replacing each value for which a predicate returns true with another value.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

#### Returns

The end of the output sequence, `__result+(__last-__first)`.

Copies each element in the range `[__first,__last)` to the range `[__result,__result+(__last-__first))` replacing elements for which `__pred` returns true with `__new_value`.

Definition at line 3171 of file `stl_algo.h`.

### 3.51.2.23 replace\_if()

```
template<typename _ForwardIterator , typename _Predicate , typename _Tp >
void std::replace_if (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    const _Tp & __new_value )
```

Replace each value in a sequence for which a predicate returns true with another value.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

**Returns**

`replace_if()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `__pred(*i)` is true then the assignment `*i = __new_value` is performed.

Definition at line 4394 of file `stl_algo.h`.

**3.51.2.24 reverse()**

```
template<typename _BidirectionalIterator >
void std::reverse (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Reverse a sequence.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.

**Returns**

`reverse()` returns no value.

Reverses the order of the elements in the range `[__first,__last)`, so that the first element becomes the last etc. For every `i` such that  $0 \leq i \leq (\_\text{last} - \_\text{first})/2$ , `reverse()` swaps `*(__first+i)` and `*(__last-(i+1))`

Definition at line 1180 of file `stl_algo.h`.

**3.51.2.25 reverse\_copy()**

```
template<typename _BidirectionalIterator , typename _OutputIterator >
_OutputIterator std::reverse_copy (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _OutputIterator __result )
```

Copy a sequence, reversing its elements.

**Parameters**

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	An output iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[__first, __last)` to the range `[__result, __result+(__last-__first))` such that the order of the elements is reversed. For every `i` such that  $0 \leq i < (\text{__last} - \text{__first})$ , `reverse_copy()` performs the assignment `*(__result+(__last-__first)-1-i) = *(__first+i)`. The ranges `[__first, __last)` and `[__result, __result+(__last-__first))` must not overlap.

Definition at line 1207 of file `stl_algo.h`.

**3.51.2.26 rotate()**

```
template<typename _ForwardIterator >
_FForwardIterator std::_V2::rotate (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last ) [inline]
```

Rotate the elements of a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

`first + (last - middle)`.

Rotates the elements of the range `[__first, __last)` by `(__middle - __first)` positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range `[__first, __last)`.

This effectively swaps the ranges `[__first, __middle)` and `[__middle, __last)`.

Performs `*(__first+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0, __last-__first)`.

Definition at line 1434 of file `stl_algo.h`.

**3.51.2.27 rotate\_copy()**

```
template<typename _ForwardIterator , typename _OutputIterator >
_OutputIterator std::rotate_copy (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last,
    _OutputIterator __result ) [inline]
```

Copy a sequence, rotating its elements.

## Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__result</code>	An output iterator.

## Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[__first,__last)` to the range beginning at

## Returns

, rotating the copied elements by `(__middle-__first)` positions so that the element at `__middle` is moved to  $\leftrightarrow$  `__result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range `[__first,__last)`.

Performs  $*(__result+(n+(__last-__middle))\%(__last-__first))=*(__first+n)$  for each `n` in the range `[0,__last-__first)`.

Definition at line 1471 of file `stl_algo.h`.

3.51.2.28 `shuffle()`

```
template<typename _RandomAccessIterator , typename _UniformRandomNumberGenerator >
void std::shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _UniformRandomNumberGenerator && __g )
```

Shuffle the elements of a sequence using a uniform random number generator.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__g</code>	A <code>UniformRandomNumberGenerator</code> (26.5.1.3).

## Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__g` to provide random numbers.

Definition at line 3792 of file `stl_algo.h`.



### 3.51.2.29 `stable_partition()`

```
template<typename _ForwardIterator , typename _Predicate >
_FowardIterator std::stable_partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred ) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

#### Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

#### Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__pred(x) == __pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1651 of file `stl_algo.h`.

### 3.51.2.30 `swap_ranges()`

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
_FowardIterator2 std::swap_ranges (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2 )
```

Swap the elements of two sequences.

#### Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.

#### Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 166 of file `stl_algobase.h`.

### 3.51.2.31 `transform()` [1/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >
_OutputIterator std::transform (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _UnaryOperation __unary_op )
```

Perform an operation on a sequence.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__unary_op</code>	A unary operator.

#### Returns

An output iterator equal to `__result+(__last-__first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=unary_op(*(__first+N))` for each `N` in the range `[0,__last-__first)`.

`unary_op` must not alter its argument.

Definition at line 4293 of file `stl_algo.h`.

### 3.51.2.32 `transform()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_BinaryOperation >
_OutputIterator std::transform (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _OutputIterator __result,
    _BinaryOperation __binary_op )
```

Perform an operation on corresponding elements of two sequences.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_op</code>	A binary operator.

**Returns**

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=__binary_op(*(__first1+N),*(__first2+N))` for each `N` in the range `[0,__last1-__first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4330 of file `stl_algo.h`.

**3.51.2.33 unique()** [1/2]

```
template<typename _ForwardIterator >
_FForwardIterator std::unique (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Remove consecutive duplicate values from a sequence.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 995 of file `stl_algo.h`.

3.51.2.34 `unique()` [2/2]

```
template<typename _ForwardIterator , typename _BinaryPredicate >
_FForwardIterator std::unique (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _BinaryPredicate __binary_pred ) [inline]
```

Remove consecutive values from a sequence using a predicate.

## Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

## Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1025 of file `stl_algo.h`.

3.51.2.35 `unique_copy()` [1/2]

```
template<typename _InputIterator , typename _OutputIterator >
_OutputIterator std::unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result ) [inline]
```

Copy a sequence, removing consecutive duplicate values.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

## Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4493 of file `stl_algo.h`.

### 3.51.2.36 `unique_copy()` [2/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
_OutputIterator std::unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred ) [inline]
```

Copy a sequence, removing consecutive values using a predicate.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_pred</code>	A binary predicate.

#### Returns

An iterator designating the end of the resulting sequence.

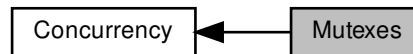
Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4534 of file `stl_algo.h`.

## 3.52 Mutexes

Collaboration diagram for Mutexes:



### Classes

- struct `std::adopt_lock_t`
- struct `std::defer_lock_t`
- class `std::lock_guard<_Mutex>`
- class `std::mutex`
- struct `std::try_to_lock_t`
- class `std::unique_lock<_Mutex>`

### Functions

- template<typename `_Mutex`>  
void `std::swap` (`unique_lock<_Mutex>` &\_\_x, `unique_lock<_Mutex>` &\_\_y) noexcept

### Variables

- `_GLIBCXX17_INLINE` constexpr `adopt_lock_t` `std::adopt_lock`
- `_GLIBCXX17_INLINE` constexpr `defer_lock_t` `std::defer_lock`
- `_GLIBCXX17_INLINE` constexpr `try_to_lock_t` `std::try_to_lock`
- template<typename `_Mutex`>  
void `std::swap` (`shared_lock<_Mutex>` &\_\_x, `shared_lock<_Mutex>` &\_\_y) noexcept
- `#define` `__cpp_lib_shared_timed_mutex`
- using `std::__shared_timed_mutex_base` = `__shared_mutex_cv`

#### 3.52.1 Detailed Description

Classes for mutex support.

#### 3.52.2 Macro Definition Documentation

### 3.52.2.1 `__cpp_lib_shared_timed_mutex`

```
#define __cpp_lib_shared_timed_mutex
```

Swap specialization for `shared_lock`.

Definition at line 57 of file `shared_mutex`.

### 3.52.3 Typedef Documentation

#### 3.52.3.1 `__shared_timed_mutex_base`

```
using std::__shared_timed_mutex_base = typedef __shared_mutex_cv
```

Swap specialization for `shared_lock`.

Definition at line 355 of file `shared_mutex`.

### 3.52.4 Function Documentation

#### 3.52.4.1 `swap()` [1/2]

```
template<typename _Mutex >  
void std::swap (  
    unique_lock< _Mutex > & __x,  
    unique_lock< _Mutex > & __y ) [inline], [noexcept]
```

Swap overload for `unique_lock` objects.

Definition at line 363 of file `std_mutex.h`.

#### 3.52.4.2 `swap()` [2/2]

```
template<typename _Mutex >  
void std::swap (  
    shared_lock< _Mutex > & __x,  
    shared_lock< _Mutex > & __y ) [noexcept]
```

Swap specialization for `shared_lock`.

Definition at line 676 of file `shared_mutex`.

### 3.52.5 Variable Documentation

#### 3.52.5.1 adopt\_lock

```
_GLIBCXX17_INLINE constexpr adopt_lock_t std::adopt_lock
```

Tag used to make a scoped lock take ownership of a locked mutex.

Definition at line 148 of file std\_mutex.h.

#### 3.52.5.2 defer\_lock

```
_GLIBCXX17_INLINE constexpr defer_lock_t std::defer_lock
```

Tag used to prevent a scoped lock from acquiring ownership of a mutex.

Definition at line 142 of file std\_mutex.h.

#### 3.52.5.3 try\_to\_lock

```
_GLIBCXX17_INLINE constexpr try_to_lock_t std::try_to_lock
```

Tag used to prevent a scoped lock from blocking if a mutex is locked.

Definition at line 145 of file std\_mutex.h.



### 3.53 Negators

Collaboration diagram for Negators:



#### Classes

- class `std::binary_negate<_Predicate>`
- class `std::unary_negate<_Predicate>`

#### Functions

- `template<typename _Predicate>`  
`_GLIBCXX14_CONSTEXPR` `unary_negate<_Predicate>` `std::not1` (`const _Predicate &__pred`)
- `template<typename _Predicate>`  
`_GLIBCXX14_CONSTEXPR` `binary_negate<_Predicate>` `std::not2` (`const _Predicate &__pred`)

#### 3.53.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};
std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (`i`) of `v` for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

#### 3.53.2 Function Documentation

### 3.53.2.1 not1()

```
template<typename _Predicate >
_GLIBCXX14_CONSTEXPR unary_negate<_Predicate> std::not1 (
    const _Predicate & __pred ) [inline]
```

One of the [negation functors](#).

Definition at line 1000 of file stl\_function.h.

### 3.53.2.2 not2()

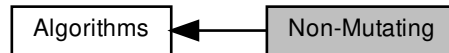
```
template<typename _Predicate >
_GLIBCXX14_CONSTEXPR binary_negate<_Predicate> std::not2 (
    const _Predicate & __pred ) [inline]
```

One of the [negation functors](#).

Definition at line 1028 of file stl\_function.h.

### 3.54 Non-Mutating

Collaboration diagram for Non-Mutating:



#### Functions

- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵  
binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp  
&__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __last, _↵  
Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2 >`  
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵  
first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵  
first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
ForwardIterator __last2)`

- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, ↵`  
`_ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`_Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵`  
`_BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵`  
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵`  
`_ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp`  
`&__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp`  
`&__val, _BinaryPredicate __binary_pred)`

### 3.54.1 Detailed Description

### 3.54.2 Function Documentation

**3.54.2.1 adjacent\_find()** [1/2]

```
template<typename _ForwardIterator >
_FowardIterator std::adjacent_find (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Find two adjacent values in a sequence that are equal.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

**Returns**

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `*i == *(i+1)`, or `__last` if no such iterator exists.

Definition at line 4024 of file `stl_algo.h`.

**3.54.2.2 adjacent\_find()** [2/2]

```
template<typename _ForwardIterator , typename _BinaryPredicate >
_FowardIterator std::adjacent_find (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _BinaryPredicate __binary_pred ) [inline]
```

Find two adjacent values in a sequence using a predicate.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

**Returns**

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `__binary_pred(*i,*(i+1))` is true, or `__last` if no such iterator exists.

Definition at line 4049 of file `stl_algo.h`.

3.54.2.3 `all_of()`

```
template<typename _InputIterator , typename _Predicate >
bool std::all_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is true for all the elements of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if `__pred` is true for each element in the range `[__first,__last)`, and false otherwise.

Definition at line 508 of file `stl_algo.h`.

References `std::find_if_not()`.

3.54.2.4 `any_of()`

```
template<typename _InputIterator , typename _Predicate >
bool std::any_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is false for at least an element of a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

## Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range `[__first,__last)` such that `__pred` is true, and false otherwise.

Definition at line 543 of file `stl_algo.h`.

References `std::none_of()`.

#### 3.54.2.5 `count()`

```
template<typename _InputIterator , typename _Tp >
iterator_traits<_InputIterator>::difference_type std::count (
    _InputIterator __first,
    _InputIterator __last,
    const _Tp & __value ) [inline]
```

Count the number of copies of a value in a sequence.

##### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be counted.

##### Returns

The number of iterators `i` in the range `[__first,__last)` for which `*i == __value`

Definition at line 4074 of file `stl_algo.h`.

#### 3.54.2.6 `count_if()`

```
template<typename _InputIterator , typename _Predicate >
iterator_traits<_InputIterator>::difference_type std::count_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Count the elements of a sequence for which a predicate is true.

##### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

The number of iterators `i` in the range `[__first,__last)` for which `__pred(*i)` is true.

Definition at line 4097 of file `stl_algo.h`.

**3.54.2.7 equal()** [1/4]

```
template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
bool std::equal (
    _IIter1 __first1,
    _IIter1 __last1,
    _IIter2 __first2,
    _BinaryPredicate __binary_pred ) [inline]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

**Returns**

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1071 of file `stl_algobase.h`.

**3.54.2.8 equal()** [2/4]

```
template<typename _II1 , typename _II2 >
bool std::equal (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2 ) [inline]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.



**Returns**

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1039 of file `stl_algobase.h`.

**3.54.2.9 `equal()`** [3/4]

```
template<typename _II1 , typename _II2 >
bool std::equal (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2 ) [inline]
```

Tests a range for element-wise equality.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1158 of file `stl_algobase.h`.

**3.54.2.10 `equal()`** [4/4]

```
template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
bool std::equal (
    _IIter1 __first1,
    _IIter1 __last1,
    _IIter2 __first2,
    _IIter2 __last2,
    _BinaryPredicate __binary_pred ) [inline]
```

Tests a range for element-wise equality.

## Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

## Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1190 of file `stl_algobase.h`.

Referenced by `std::operator==()`.

3.54.2.11 `find()`

```
template<typename _InputIterator , typename _Tp >
_InputIterator std::find (
    _InputIterator __first,
    _InputIterator __last,
    const _Tp & __val ) [inline]
```

Find the first occurrence of a value in a sequence.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__val</code>	The value to find.

## Returns

The first iterator `i` in the range `[__first,__last)` such that `*i == __val`, or `__last` if no such iterator exists.

Definition at line 3897 of file `stl_algo.h`.

3.54.2.12 `find_end()` [1/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
_FForwardIterator1 std::find_end (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2 ) [inline]
```

Find last matching subsequence in a sequence.

## Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.

## Returns

The last iterator `i` in the range `[__first1, __last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the `__first` element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first1, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1-(__last2-__first2))`.

Definition at line 425 of file `stl_algo.h`.

3.54.2.13 `find_end()` [2/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
_FForwardIterator1 std::find_end (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __comp ) [inline]
```

Find last matching subsequence in a sequence using a predicate.

## Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.
<code>__comp</code>	The predicate to use.

**Returns**

The last iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `predicate(*(i+N), (__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

Definition at line 474 of file `stl_algo.h`.

**3.54.2.14 find\_first\_of() [1/2]**

```
template<typename _InputIterator , typename _ForwardIterator >
_InputIterator std::find_first_of (
    _InputIterator __first1,
    _InputIterator __last1,
    _ForwardIterator __first2,
    _ForwardIterator __last2 )
```

Find element from a set in a sequence.

**Parameters**

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.

**Returns**

The first iterator `i` in the range `[__first1, __last1)` such that `*i == *(i2)` such that `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 3952 of file `stl_algo.h`.

**3.54.2.15 find\_first\_of()** [2/2]

```
template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
_InputIterator std::find_first_of (
    _InputIterator __first1,
    _InputIterator __last1,
    _ForwardIterator __first2,
    _ForwardIterator __last2,
    _BinaryPredicate __comp )
```

Find element from a set in a sequence using a predicate.

**Parameters**

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.
<code>__comp</code>	Predicate to use.

**Returns**

The first iterator `i` in the range `[__first1,__last1)` such that `comp(*i,*(i2))` is true and `i2` is an iterator in `[__first2,↵__last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1,__last1)` for an element that is equal to some element in the range `[__first2,__last2)`. If found, returns an iterator in the range `[__first1,__last1)`, otherwise returns `__last1`.

Definition at line 3993 of file `stl_algo.h`.

**3.54.2.16 find\_if()**

```
template<typename _InputIterator , typename _Predicate >
_InputIterator std::find_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is true.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

The first iterator `i` in the range `[__first,__last)` such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 3921 of file `stl_algo.h`.

Referenced by `std::none_of()`.

**3.54.2.17 find\_if\_not()**

```
template<typename _InputIterator , typename _Predicate >
_InputIterator std::find_if_not (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is false.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

The first iterator `i` in the range `[__first,__last)` such that `__pred(*i)` is false, or `__last` if no such iterator exists.

Definition at line 558 of file `stl_algo.h`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

**3.54.2.18 for\_each()**

```
template<typename _InputIterator , typename _Function >
_Function std::for_each (
    _InputIterator __first,
    _InputIterator __last,
    _Function __f )
```

Apply a function to every element of a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__f</code>	A unary function object.

**Returns**

`__f`

Applies the function object `__f` to each element in the range `[first,last)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

Definition at line 3876 of file `stl_algo.h`.

**3.54.2.19 is\_permutation()** [1/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2 ) [inline]
```

Checks whether a permutation of the second sequence is equal to the first sequence.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

**Returns**

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 3542 of file `stl_algo.h`.

**3.54.2.20 is\_permutation()** [2/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
bool std::is_permutation (
    _ForwardIterator1 __first1,
```

```
_ForwardIterator1 __last1,  
_ForwardIterator2 __first2,  
_BinaryPredicate __pred ) [inline]
```

Checks whether a permutation of the second sequence is equal to the first sequence.



**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__pred</code>	A binary predicate.

**Returns**

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` `begin`, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 3574 of file `stl_algo.h`.

**3.54.2.21 is\_permutation()** [3/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2 ) [inline]
```

Checks whether a permutaion of the second sequence is equal to the first sequence.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of first range.

**Returns**

true if there exists a permutation of the elements in the range `[__first2, __last2)`, beginning with `ForwardIterator2` `begin`, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 3666 of file `stl_algo.h`.

**3.54.2.22 is\_permutation()** [4/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
bool std::is_permutation (
```

```

_FowardIterator1 __first1,
_FowardIterator1 __last1,
_FowardIterator2 __first2,
_FowardIterator2 __last2,
_BinaryPredicate __pred ) [inline]

```

Checks whether a permutation of the second sequence is equal to the first sequence.

#### Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of first range.
<code>__pred</code>	A binary predicate.

#### Returns

true if there exists a permutation of the elements in the range `[__first2, __last2)`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 3694 of file `stl_algo.h`.

#### 3.54.2.23 mismatch() [1/4]

```

template<typename _InputIterator1 , typename _InputIterator2 >
pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2 ) [inline]

```

Finds the places in ranges which don't match.

#### Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

#### Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1300 of file `stl_algobase.h`.

**3.54.2.24 mismatch()** [2/4]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _BinaryPredicate __binary_pred ) [inline]
```

Finds the places in ranges which don't match.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

**Returns**

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1334 of file `stl_algobase.h`.

**3.54.2.25 mismatch()** [3/4]

```
template<typename _InputIterator1 , typename _InputIterator2 >
pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2 ) [inline]
```

Finds the places in ranges which don't match.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1380 of file `stl_algobase.h`.

**3.54.2.26 mismatch()** [4/4]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _BinaryPredicate __binary_pred ) [inline]
```

Finds the places in ranges which don't match.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate <a href="#">functor</a> .

**Returns**

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1416 of file `stl_algobase.h`.

**3.54.2.27 none\_of()**

```
template<typename _InputIterator , typename _Predicate >
bool std::none_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is false for all the elements of a sequence.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

**Returns**

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range `[__first,__last)`, and false otherwise.

Definition at line 525 of file `stl_algo.h`.

References `std::find_if()`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

**3.54.2.28 search()** [1/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
_FForwardIterator1 std::search (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2 ) [inline]
```

Search a sequence for a matching sub-sequence.

**Parameters**

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.

**Returns**

The first iterator `i` in the range `[__first1,__last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0,__last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1,__last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2,__last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[__first1,__last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

Definition at line 4137 of file `stl_algo.h`.

#### 3.54.2.29 `search()` [2/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
_FForwardIterator1 std::search (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __predicate ) [inline]
```

Search a sequence for a matching sub-sequence using a predicate.

##### Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.
<code>__predicate</code>	A binary predicate.

##### Returns

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `__predicate(*(i+N),*(__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)`, using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

##### See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4177 of file `stl_algo.h`.

#### 3.54.2.30 `search_n()` [1/2]

```
template<typename _ForwardIterator , typename _Integer , typename _Tp >
_FForwardIterator std::search_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    const _Tp & __val ) [inline]
```

Search a sequence for a number of consecutive values.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.

**Returns**

The first iterator `i` in the range `[__first, __last-__count)` such that `*(i+N) == __val` for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `count` consecutive elements equal to `__val`.

Definition at line 4211 of file `stl_algo.h`.

**3.54.2.31 search\_n() [2/2]**

```
template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate
>
_FForwardIterator std::search_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    const _Tp & __val,
    _BinaryPredicate __binary_pred ) [inline]
```

Search a sequence for a number of consecutive values using a predicate.

**Parameters**

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.
<code>__binary_pred</code>	A binary predicate.

**Returns**

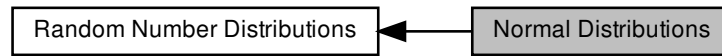
The first iterator `i` in the range `[__first, __last-__count)` such that `__binary_pred(*(i+N), __val)` is true for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `__count` consecutive elements for which the predicate returns true.

Definition at line 4245 of file `stl_algo.h`.

## 3.55 Normal Distributions

Collaboration diagram for Normal Distributions:



## Classes

- class `std::cauchy_distribution<_RealType>`
- class `std::chi_squared_distribution<_RealType>`
- class `std::fisher_f_distribution<_RealType>`
- class `std::gamma_distribution<_RealType>`
- class `std::lognormal_distribution<_RealType>`
- class `std::normal_distribution<_RealType>`
- class `std::student_t_distribution<_RealType>`

## Functions

- template<typename \_RealType>  
bool `std::operator!=` (const `std::normal_distribution<_RealType>` &\_\_d1, const `std::normal_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::lognormal_distribution<_RealType>` &\_\_d1, const `std::lognormal_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::gamma_distribution<_RealType>` &\_\_d1, const `std::gamma_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::chi_squared_distribution<_RealType>` &\_\_d1, const `std::chi_squared_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::cauchy_distribution<_RealType>` &\_\_d1, const `std::cauchy_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::fisher_f_distribution<_RealType>` &\_\_d1, const `std::fisher_f_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType>  
bool `std::operator!=` (const `std::student_t_distribution<_RealType>` &\_\_d1, const `std::student_t_distribution<_RealType>` &\_\_d2)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &\_\_os, const `std::cauchy_distribution<_RealType>` &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_istream<_CharT, _Traits>` & `std::operator>>` (`std::basic_istream<_CharT, _Traits>` &\_\_is, `std::cauchy_distribution<_RealType>` &\_\_x)



### 3.55.1 Detailed Description

### 3.55.2 Function Documentation

#### 3.55.2.1 `operator!=()` [1/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::normal_distribution< _RealType > & __d1,
    const std::normal_distribution< _RealType > & __d2 ) [inline]
```

Return true if two normal distributions are different.

Definition at line 2128 of file random.h.

#### 3.55.2.2 `operator!=()` [2/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::lognormal_distribution< _RealType > & __d1,
    const std::lognormal_distribution< _RealType > & __d2 ) [inline]
```

Return true if two lognormal distributions are different.

Definition at line 2337 of file random.h.

#### 3.55.2.3 `operator!=()` [3/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::gamma_distribution< _RealType > & __d1,
    const std::gamma_distribution< _RealType > & __d2 ) [inline]
```

Return true if two gamma distributions are different.

Definition at line 2562 of file random.h.

**3.55.2.4 operator!=( )** [4/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::chi_squared_distribution< _RealType > & __d1,
    const std::chi_squared_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Chi-squared distributions are different.

Definition at line 2782 of file random.h.

**3.55.2.5 operator!=( )** [5/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::cauchy_distribution< _RealType > & __d1,
    const std::cauchy_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Cauchy distributions have different parameters.

Definition at line 2954 of file random.h.

**3.55.2.6 operator!=( )** [6/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::fisher_f_distribution< _RealType > & __d1,
    const std::fisher_f_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Fisher f distributions are different.

Definition at line 3215 of file random.h.

**3.55.2.7 operator!=( )** [7/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::student_t_distribution< _RealType > & __d1,
    const std::student_t_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Student t distributions are different.

Definition at line 3433 of file random.h.

**3.55.2.8 operator<<( )**

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<<(
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::cauchy_distribution< _RealType > & __x )
```

Inserts a cauchy\_distribution random number distribution \_\_x into the output stream \_\_os.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A <code>cauchy_distribution</code> random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2130 of file `bits/random.tcc`.

References `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::ios_base::precision()`, `std::scientific()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

**3.55.2.9 operator>>()**

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::cauchy_distribution< _RealType > & __x )
```

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>cauchy_distribution</code> random number generator engine.

**Returns**

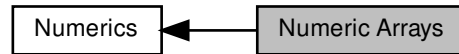
The input stream with `__x` extracted or in an error state.

Definition at line 2154 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType >::param()`, and `std::skipws()`.

## 3.56 Numeric Arrays

Collaboration diagram for Numeric Arrays:



## Classes

- class `std::gslice`
- class `std::gslice_array< _Tp >`
- class `std::indirect_array< _Tp >`
- class `std::mask_array< _Tp >`
- class `std::slice`
- class `std::slice_array< _Tp >`
- class `std::valarray< _Tp >`

## Macros

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

## Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array< _Tp >::gslice_array (const gslice_array &)`
- `std::indirect_array< _Tp >::indirect_array (const indirect_array &)`
- `std::mask_array< _Tp >::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t __o, size_t __d, size_t __s)`
- `std::slice_array< _Tp >::slice_array (const slice_array &)`
- `std::valarray< _Tp >::valarray ()`
- `std::valarray< _Tp >::valarray (size_t)`

- `std::valarray<_Tp>::valarray` (const `_Tp` &, `size_t`)
- `std::valarray<_Tp>::valarray` (const `valarray` &)
- `std::valarray<_Tp>::valarray` (`valarray` &&) noexcept
- `std::valarray<_Tp>::valarray` (const `slice_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (const `gslice_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (const `mask_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (const `indirect_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (`initializer_list<_Tp>`)
- `template<class _Dom>`  
`std::valarray<_Tp>::valarray` (const `_Expr<_Dom, _Tp>` & `__e`)
- `template<typename _Tp>`  
`std::valarray<_Tp>::valarray` (const `_Tp *` `__restrict` `__p`, `size_t` `__n`)
- `std::gslice::~gslice` ()
- `_Expr<_ValFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply` (`_Tp` `func(_Tp)`) const
- `_Expr<_RefFunClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply` (`_Tp` `func(const _Tp &)`) const
- `template<class _Tp>`  
`_Tp *` `std::begin` (`valarray<_Tp>` & `__va`)
- `template<class _Tp>`  
`const _Tp *` `std::begin` (const `valarray<_Tp>` & `__va`)
- `valarray<_Tp> std::valarray<_Tp>::cshift` (int `__n`) const
- `template<class _Tp>`  
`_Tp *` `std::end` (`valarray<_Tp>` & `__va`)
- `template<class _Tp>`  
`const _Tp *` `std::end` (const `valarray<_Tp>` & `__va`)
- `_Tp std::valarray<_Tp>::max` () const
- `_Tp std::valarray<_Tp>::min` () const
- `_UnaryOp<__logical_not>::Rt std::valarray<_Tp>::operator!` () const
- `template<typename _Tp>`  
`_Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type> std::operator!=` (const `valarray<_Tp>` & `__v`, const `_Tp` & `__t`)
- `template<typename _Tp>`  
`_Expr<_BinClos<__not_equal_to, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type> std::operator!=` (const `_Tp` & `__t`, const `valarray<_Tp>` & `__v`)
- `template<typename _Tp>`  
`_Expr<_BinClos<__not_equal_to, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type> std::operator!=` (const `valarray<_Tp>` & `__v`, const `valarray<_Tp>` & `__w`)
- `template<typename _Tp>`  
`_Expr<_BinClos<__modulus, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__modulus, _Tp>::result_type> std::operator%` (const `valarray<_Tp>` & `__v`, const `valarray<_Tp>` & `__w`)
- `template<typename _Tp>`  
`_Expr<_BinClos<__modulus, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__modulus, _Tp>::result_type> std::operator%` (const `valarray<_Tp>` & `__v`, const `_Tp` & `__t`)
- `template<typename _Tp>`  
`_Expr<_BinClos<__modulus, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__modulus, _Tp>::result_type> std::operator%` (const `_Tp` & `__t`, const `valarray<_Tp>` & `__v`)
- `void std::gslice_array<_Tp>::operator%=` (const `valarray<_Tp>` & `__v`) const
- `void std::mask_array<_Tp>::operator%=` (const `valarray<_Tp>` & `__v`) const
- `void std::indirect_array<_Tp>::operator%=` (const `valarray<_Tp>` & `__v`) const
- `template<class _Dom>`  
`void std::gslice_array<_Tp>::operator%=` (const `_Expr<_Dom, _Tp>` & `__e`) const
- `template<class _Dom>`  
`void std::indirect_array<_Tp>::operator%=` (const `_Expr<_Dom, _Tp>` & `__e`) const

- `template<class _Dom >`  
`void std::mask_array<_Tp>::operator%=(const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator%=(const valarray<_Tp> &) const`
- `template<class _Dom >`  
`void std::slice_array<_Tp>::operator%=(const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator%=(const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator%=(const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & std::valarray<_Tp>::operator%=(const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_and, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__bitwise_and, _Tp>::result_type> std::operator&(const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_and, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__bitwise_and, _Tp>::result_type> std::operator&(const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__bitwise_and, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__bitwise_and, _Tp>::result_type> std::operator&(const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__logical_and, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__logical_and, _Tp>::result_type> std::operator&&(const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__logical_and, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__logical_and, _Tp>::result_type> std::operator&&(const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__logical_and, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__logical_and, _Tp>::result_type> std::operator&&(const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `void std::gslice_array<_Tp>::operator&=(const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator&=(const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator&=(const valarray<_Tp> &) const`
- `template<class _Dom >`  
`void std::gslice_array<_Tp>::operator&=(const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`  
`void std::indirect_array<_Tp>::operator&=(const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`  
`void std::mask_array<_Tp>::operator&=(const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator&=(const valarray<_Tp> &) const`
- `template<class _Dom >`  
`void std::slice_array<_Tp>::operator&=(const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator&=(const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator&=(const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & std::valarray<_Tp>::operator&=(const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__multiplies, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__multiplies, _Tp>::result_type> std::operator*(const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__multiplies, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__multiplies, _Tp>::result_type> std::operator*(const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`  
`_Expr<_BinClos<__multiplies, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__multiplies, _Tp>::result_type> std::operator*(const valarray<_Tp> &__v, const valarray<_Tp> &__w)`

- void `std::gslice_array<_Tp>::operator*=` (const `valarray<_Tp>` &) const
- void `std::mask_array<_Tp>::operator*=` (const `valarray<_Tp>` &) const
- void `std::indirect_array<_Tp>::operator*=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `std::gslice_array<_Tp>::operator*=` (const `_Expr<_Dom, _Tp>` &) const
- template<class `_Dom`>  
void `std::indirect_array<_Tp>::operator*=` (const `_Expr<_Dom, _Tp>` &) const
- template<class `_Dom`>  
void `std::mask_array<_Tp>::operator*=` (const `_Expr<_Dom, _Tp>` &) const
- void `std::slice_array<_Tp>::operator*=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `std::slice_array<_Tp>::operator*=` (const `_Expr<_Dom, _Tp>` &) const
- `valarray<_Tp>` & `std::valarray<_Tp>::operator*=` (const `_Tp` &)
- `valarray<_Tp>` & `std::valarray<_Tp>::operator*=` (const `valarray<_Tp>` &)
- template<class `_Dom`>  
`valarray<_Tp>` & `std::valarray<_Tp>::operator*=` (const `_Expr<_Dom, _Tp>` &)
- `_UnaryOp<__unary_plus>::Rt std::valarray<_Tp>::operator+` () const
- template<typename `_Tp`>  
`_Expr<__BinClos<__plus, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__plus, _Tp>::result_type>`  
`std::operator+` (const `valarray<_Tp>` & `_v`, const `_Tp` & `_t`)
- template<typename `_Tp`>  
`_Expr<__BinClos<__plus, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__plus, _Tp>::result_type>`  
`std::operator+` (const `valarray<_Tp>` & `_v`, const `valarray<_Tp>` & `_w`)
- template<typename `_Tp`>  
`_Expr<__BinClos<__plus, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__plus, _Tp>::result_type>`  
`std::operator+` (const `_Tp` & `_t`, const `valarray<_Tp>` & `_v`)
- void `std::gslice_array<_Tp>::operator+=` (const `valarray<_Tp>` &) const
- void `std::mask_array<_Tp>::operator+=` (const `valarray<_Tp>` &) const
- void `std::indirect_array<_Tp>::operator+=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `std::gslice_array<_Tp>::operator+=` (const `_Expr<_Dom, _Tp>` &) const
- template<class `_Dom`>  
void `std::indirect_array<_Tp>::operator+=` (const `_Expr<_Dom, _Tp>` &) const
- template<class `_Dom`>  
void `std::mask_array<_Tp>::operator+=` (const `_Expr<_Dom, _Tp>` &) const
- void `std::slice_array<_Tp>::operator+=` (const `valarray<_Tp>` &) const
- template<class `_Dom`>  
void `std::slice_array<_Tp>::operator+=` (const `_Expr<_Dom, _Tp>` &) const
- `valarray<_Tp>` & `std::valarray<_Tp>::operator+=` (const `_Tp` &)
- `valarray<_Tp>` & `std::valarray<_Tp>::operator+=` (const `valarray<_Tp>` &)
- template<class `_Dom`>  
`valarray<_Tp>` & `std::valarray<_Tp>::operator+=` (const `_Expr<_Dom, _Tp>` &)
- `_UnaryOp<__negate>::Rt std::valarray<_Tp>::operator-` () const
- template<typename `_Tp`>  
`_Expr<__BinClos<__minus, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__minus, _Tp>::result_type>`  
`std::operator-` (const `valarray<_Tp>` & `_v`, const `valarray<_Tp>` & `_w`)
- template<typename `_Tp`>  
`_Expr<__BinClos<__minus, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__minus, _Tp>::result_type>`  
`std::operator-` (const `valarray<_Tp>` & `_v`, const `_Tp` & `_t`)
- template<typename `_Tp`>  
`_Expr<__BinClos<__minus, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__minus, _Tp>::result_type>`  
`std::operator-` (const `_Tp` & `_t`, const `valarray<_Tp>` & `_v`)

- void `std::gslice_array<_Tp>::operator=` (const `valarray<_Tp>` &) const
- void `std::mask_array<_Tp>::operator=` (const `valarray<_Tp>` &) const
- void `std::indirect_array<_Tp>::operator=` (const `valarray<_Tp>` &) const
- template<class \_Dom >  
void `std::gslice_array<_Tp>::operator=` (const `_Expr<_Dom, _Tp>` &) const
- template<class \_Dom >  
void `std::indirect_array<_Tp>::operator=` (const `_Expr<_Dom, _Tp>` &) const
- template<class \_Dom >  
void `std::mask_array<_Tp>::operator=` (const `_Expr<_Dom, _Tp>` &) const
- void `std::slice_array<_Tp>::operator=` (const `valarray<_Tp>` &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator=` (const `_Expr<_Dom, _Tp>` &) const
- `valarray<_Tp>` & `std::valarray<_Tp>::operator=` (const `_Tp` &)
- `valarray<_Tp>` & `std::valarray<_Tp>::operator=` (const `valarray<_Tp>` &)
- template<class \_Dom >  
`valarray<_Tp>` & `std::valarray<_Tp>::operator=` (const `_Expr<_Dom, _Tp>` &)
- template<typename \_Tp >  
`_Expr<_BinClos<__divides, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__divides, _Tp>::result_type > std::operator/` (const `valarray<_Tp>` &\_\_v, const `valarray<_Tp>` &\_\_w)
- template<typename \_Tp >  
`_Expr<_BinClos<__divides, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__divides, _Tp>::result_type > std::operator/` (const `valarray<_Tp>` &\_\_v, const `_Tp` &\_\_t)
- template<typename \_Tp >  
`_Expr<_BinClos<__divides, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__divides, _Tp>::result_type > std::operator/` (const `_Tp` &\_\_t, const `valarray<_Tp>` &\_\_v)
- void `std::gslice_array<_Tp>::operator/=` (const `valarray<_Tp>` &) const
- void `std::mask_array<_Tp>::operator/=` (const `valarray<_Tp>` &) const
- void `std::indirect_array<_Tp>::operator/=` (const `valarray<_Tp>` &) const
- template<class \_Dom >  
void `std::gslice_array<_Tp>::operator/=` (const `_Expr<_Dom, _Tp>` &) const
- template<class \_Dom >  
void `std::mask_array<_Tp>::operator/=` (const `_Expr<_Dom, _Tp>` &) const
- template<class \_Dom >  
void `std::indirect_array<_Tp>::operator/=` (const `_Expr<_Dom, _Tp>` &) const
- void `std::slice_array<_Tp>::operator/=` (const `valarray<_Tp>` &) const
- template<class \_Dom >  
void `std::slice_array<_Tp>::operator/=` (const `_Expr<_Dom, _Tp>` &) const
- `valarray<_Tp>` & `std::valarray<_Tp>::operator/=` (const `_Tp` &)
- `valarray<_Tp>` & `std::valarray<_Tp>::operator/=` (const `valarray<_Tp>` &)
- template<class \_Dom >  
`valarray<_Tp>` & `std::valarray<_Tp>::operator/=` (const `_Expr<_Dom, _Tp>` &)
- template<typename \_Tp >  
`_Expr<_BinClos<__less, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__less, _Tp>::result_type > std::operator<` (const `valarray<_Tp>` &\_\_v, const `valarray<_Tp>` &\_\_w)
- template<typename \_Tp >  
`_Expr<_BinClos<__less, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__less, _Tp>::result_type > std::operator<` (const `valarray<_Tp>` &\_\_v, const `_Tp` &\_\_t)
- template<typename \_Tp >  
`_Expr<_BinClos<__less, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__less, _Tp>::result_type > std::operator<` (const `_Tp` &\_\_t, const `valarray<_Tp>` &\_\_v)
- template<typename \_Tp >  
`_Expr<_BinClos<__shift_left, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__shift_left, _Tp>::result_type > std::operator<<` (const `_Tp` &\_\_t, const `valarray<_Tp>` &\_\_v)



- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_t >`  
`_type > std::operator<< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_t >`  
`_type > std::operator<< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice\_array< _Tp >::operator<=<= (const valarray< _Tp > &) const`
- `void std::mask\_array< _Tp >::operator<=<= (const valarray< _Tp > &) const`
- `void std::indirect\_array< _Tp >::operator<=<= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array< _Tp >::operator<=<= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::indirect_array< _Tp >::operator<=<= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::mask_array< _Tp >::operator<=<= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice\_array< _Tp >::operator<=<= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::slice_array< _Tp >::operator<=<= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator<=<= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator<=<= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray< _Tp >::operator<=<= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type >`  
`::result_type > std::operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type >`  
`::result_type > std::operator<= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type >`  
`::result_type > std::operator<= (const valarray< _Tp > &__v, const _Tp &__t)`
- `gslice\_array & std::gslice_array< _Tp >::operator= (const gslice\_array &)`
- `indirect\_array & std::indirect_array< _Tp >::operator= (const indirect\_array &)`
- `mask\_array & std::mask_array< _Tp >::operator= (const mask\_array &)`
- `void std::gslice_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `gslice & std::gslice::operator= (const gslice &)`
- `void std::gslice_array< _Tp >::operator= (const _Tp &) const`
- `void std::mask_array< _Tp >::operator= (const _Tp &) const`
- `void std::indirect_array< _Tp >::operator= (const _Tp &) const`
- `template<class _Dom >`  
`void std::gslice_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::indirect_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `slice\_array & std::slice_array< _Tp >::operator= (const slice\_array &)`
- `void std::slice_array< _Tp >::operator= (const valarray< _Tp > &) const`
- `void std::slice_array< _Tp >::operator= (const _Tp &) const`
- `template<class _Dom >`  
`void std::slice_array< _Tp >::operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Ex >`  
`void std::mask_array< _Tp >::operator= (const _Expr< _Ex, _Tp > &__e) const`

- `valarray<_Tp> & std::valarray<_Tp>::operator= (const valarray<_Tp> &__v)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (valarray<_Tp> &&__v) noexcept`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const _Tp &__t)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const slice_array<_Tp> &__sa)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const gslice_array<_Tp> &__ga)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const mask_array<_Tp> &__ma)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const indirect_array<_Tp> &__ia)`
- `valarray & std::valarray<_Tp>::operator= (initializer_list<_Tp> __l)`
- `template<class _Dom>`  
`valarray<_Tp> & std::valarray<_Tp>::operator= (const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__equal_to, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__equal_to, _Tp>::result_type> std::operator== (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__equal_to, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__equal_to, _Tp>::result_type> std::operator== (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__equal_to, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__equal_to, _Tp>::result_type> std::operator== (const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__greater, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__greater, _Tp>::result_type> std::operator> (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__greater, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__greater, _Tp>::result_type> std::operator> (const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__greater, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__greater, _Tp>::result_type> std::operator> (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__greater_equal, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__greater_equal, _Tp>::result_type> std::operator>= (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__greater_equal, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__greater_equal, _Tp>::result_type> std::operator>= (const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__greater_equal, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__greater_equal, _Tp>::result_type> std::operator>= (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__shift_right, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__shift_right, _Tp>::result_type> std::operator>> (const valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__shift_right, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__shift_right, _Tp>::result_type> std::operator>> (const _Tp &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__shift_right, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__shift_right, _Tp>::result_type> std::operator>> (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `void std::gslice_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void std::gslice_array<_Tp>::operator>>= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom>`  
`void std::indirect_array<_Tp>::operator>>= (const _Expr<_Dom, _Tp> &) const`

- `template<class _Dom >`  
`void std::mask_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::slice_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator>>= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &)`
- `_Tp & std::valarray< _Tp >::operator[] (size_t __i)`
- `const _Tp & std::valarray< _Tp >::operator[] (size_t) const`
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (slice __s) const`
- `slice_array< _Tp > std::valarray< _Tp >::operator[] (slice __s)`
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const gslice &__s) const`
- `gslice_array< _Tp > std::valarray< _Tp >::operator[] (const gslice &__s)`
- `valarray< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m) const`
- `mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m)`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i) const`
- `indirect_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::mask_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::indirect_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::slice_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator^= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >↵`  
`::result_type > std::operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::gslice_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::indirect_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`  
`void std::mask_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void std::slice_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator|= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator|= (const valarray< _Tp > &)`
- `template<class _Dom >`  
`valarray< _Tp > & std::valarray< _Tp >::operator|= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > std::operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > std::operator|| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > std::operator|| (const _Tp &__t, const valarray< _Tp > &__v)`
- `_UnaryOp< __bitwise_not >::Rt std::valarray< _Tp >::operator~ () const`
- `void std::valarray< _Tp >::resize (size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > std::valarray< _Tp >::shift (int __n) const`
- `size_t std::slice::size () const`
- `valarray< size_t > std::gslice::size () const`
- `size_t std::valarray< _Tp >::size () const`
- `size_t std::slice::start () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::stride () const`
- `valarray< size_t > std::gslice::stride () const`
- `_Tp std::valarray< _Tp >::sum () const`
- `void std::valarray< _Tp >::swap (valarray< _Tp > &__v) noexcept`

### 3.56.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

### 3.56.2 Function Documentation

**3.56.2.1 gslice()** [1/3]

```
std::gslice::gslice ( ) [inline]
```

Construct an empty slice.

Definition at line 149 of file gslice.h.




**3.56.2.2 gslice()** [2/3]

```
std::gslice::gslice (
    size_t __o,
    const valarray< size_t > & __l,
    const valarray< size_t > & __s ) [inline]
```

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

**Parameters**

 <b>__o</b>	Offset in array of first element.
 <b>__l</b>	Array of dimension lengths.
 <b>__s</b>	Array of dimension strides between array elements.

Definition at line 153 of file gslice.h.

**3.56.2.3 gslice()** [3/3]

```
std::gslice::gslice (
    const gslice & __g ) [inline]
```

Copy constructor.

Definition at line 158 of file gslice.h.

**3.56.2.4 gslice\_array()**

```
template<typename _Tp >
std::gslice_array< _Tp >::gslice_array (
    const gslice_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file gslice\_array.h.

## 3.56.2.5 indirect\_array()

```
template<typename _Tp >
std::indirect_array< _Tp >::indirect_array (
    const indirect_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file indirect\_array.h.

## 3.56.2.6 mask\_array()

```
template<typename _Tp >
std::mask_array< _Tp >::mask_array (
    const mask_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 139 of file mask\_array.h.

## 3.56.2.7 slice() [1/2]

```
std::slice::slice ( ) [inline]
```

Construct an empty slice.

Definition at line 90 of file slice\_array.h.

## 3.56.2.8 slice() [2/2]

```
std::slice::slice (
    size_t __o,
    size_t __d,
    size_t __s ) [inline]
```

Construct a slice.

## Parameters

$\leftarrow$ __o	Offset in array of first element.
$\leftarrow$ __d	Number of elements in slice.
$\leftarrow$ __s	Stride between array elements.

Definition at line 94 of file slice\_array.h.

### 3.56.2.9 slice\_array()

```
template<typename _Tp >
std::slice_array< _Tp >::slice_array (
    const slice_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 207 of file slice\_array.h.

### 3.56.2.10 valarray() [1/10]

```
template<typename _Tp >
std::valarray< _Tp >::valarray ( ) [inline]
```

Construct an empty array.

Definition at line 610 of file valarray.

### 3.56.2.11 valarray() [2/10]

```
template<typename _Tp >
std::valarray< _Tp >::valarray (
    size_t __n ) [inline], [explicit]
```

Construct an array with  $n$  elements.

Definition at line 614 of file valarray.

### 3.56.2.12 valarray() [3/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const _Tp & __t,
    size_t __n ) [inline]
```

Construct an array with  $n$  elements initialized to  $t$ .

Definition at line 620 of file valarray.

**3.56.2.13** `valarray()` [4/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const valarray< _Tp > & __v ) [inline]
```

Copy constructor.

Definition at line 635 of file `valarray`.

**3.56.2.14** `valarray()` [5/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    valarray< _Tp > && __v ) [inline], [noexcept]
```

Move constructor.

Definition at line 643 of file `valarray`.

**3.56.2.15** `valarray()` [6/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const slice_array< _Tp > & __sa ) [inline]
```

Construct an array with the same size and values in *sa*.

Definition at line 653 of file `valarray`.

**3.56.2.16** `valarray()` [7/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const gslice_array< _Tp > & __ga ) [inline]
```

Construct an array with the same size and values in *ga*.

Definition at line 662 of file `valarray`.



**3.56.2.17 valarray()** [8/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const mask_array< _Tp > & __ma ) [inline]
```

Construct an array with the same size and values in *ma*.

Definition at line 673 of file valarray.

**3.56.2.18 valarray()** [9/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const indirect_array< _Tp > & __ia ) [inline]
```

Construct an array with the same size and values in *ia*.

Definition at line 682 of file valarray.

**3.56.2.19 valarray()** [10/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    initializer_list< _Tp > __l ) [inline]
```

Construct an array with an `initializer_list` of values.

Definition at line 692 of file valarray.

**3.56.2.20 ~gslice()**

```
std::gslice::~gslice ( ) [inline]
```

Destructor.

Definition at line 163 of file gslice.h.

**3.56.2.21 apply()** [1/2]

```
template<class _Tp>
_Expr< _ValFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
    _Tp func_Tp ) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying `func` to the corresponding element of this array. The new array has the same size as this one.

## Parameters

<i>func</i>	Function of Tp returning Tp to apply.
-------------	---------------------------------------

## Returns

New valarray with transformed elements.

Definition at line 1054 of file valarray.

3.56.2.22 `apply()` [2/2]

```
template<class _Tp>
_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
    _Tp funcconst _Tp & ) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

## Parameters

<i>func</i>	Function of const Tp& returning Tp to apply.
-------------	--

## Returns

New valarray with transformed elements.

Definition at line 1062 of file valarray.

3.56.2.23 `begin()` [1/2]

```
template<class _Tp >
_Tp * std::begin (
    valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to the first element of the valarray.

## Parameters

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1201 of file valarray.

#### 3.56.2.24 begin() [2/2]

```
template<class _Tp >
const _Tp * std::begin (
    const valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to the first element of the const valarray.

##### Parameters

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1211 of file valarray.

#### 3.56.2.25 cshift()

```
template<class _Tp >
valarray< _Tp > std::valarray< _Tp >::cshift (
    int __n ) const [inline]
```

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index  $i$ , the new position is  $(i - n) \% \text{size}()$ . The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

##### Parameters

<code>__n</code>	Number of element positions to rotate.
------------------	--

##### Returns

New valarray with elements in shifted positions.

Definition at line 980 of file valarray.

**3.56.2.26** `end()` [1/2]

```
template<class _Tp >
_Tp * std::end (
    valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to one past the last element of the valarray.

**Parameters**

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1221 of file valarray.

**3.56.2.27** `end()` [2/2]

```
template<class _Tp >
const _Tp * std::end (
    const valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to one past the last element of the const valarray.

**Parameters**

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1231 of file valarray.

**3.56.2.28** `max()`

```
template<typename _Tp >
_Tp std::valarray< _Tp >::max ( ) const [inline]
```

Return the maximum element using operator<().

Definition at line 1046 of file valarray.

**3.56.2.29** `min()`

```
template<typename _Tp >
_Tp std::valarray< _Tp >::min ( ) const [inline]
```

Return the minimum element using operator<().

Definition at line 1038 of file valarray.

**3.56.2.30 operator!()**

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __logical_not >::_Rt std::valarray< _Tp >::operator! ( )
const [inline]
```

Return a new valarray by applying unary ! to each element.

Definition at line 1081 of file valarray.

**3.56.2.31 operator%=( ) [1/6]**

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 202 of file gslice\_array.h.

**3.56.2.32 operator%=( ) [2/6]**

```
template<typename _Tp >
void std::mask_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 192 of file mask\_array.h.

**3.56.2.33 operator%=( ) [3/6]**

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 196 of file indirect\_array.h.

**3.56.2.34 operator%=( )** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 258 of file slice\_array.h.

**3.56.2.35 operator%=( )** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to *e % t*.

Definition at line 1108 of file valarray.

**3.56.2.36 operator%=( )** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
    const valarray< _Tp > & __v ) [inline]
```

Modulo elements of array by corresponding elements of *v*.

Definition at line 1108 of file valarray.

**3.56.2.37 operator&=( )** [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 206 of file gslice\_array.h.

**3.56.2.38 operator&=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 196 of file mask\_array.h.

**3.56.2.39 operator&=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 200 of file indirect\_array.h.

**3.56.2.40 operator&=()** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 262 of file slice\_array.h.

**3.56.2.41 operator&=()** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to *e* & *t*.

Definition at line 1110 of file valarray.

**3.56.2.42 operator&=()** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
    const valarray< _Tp > & __v ) [inline]
```

Logical and corresponding elements of *v* with elements of array.

Definition at line 1110 of file valarray.

**3.56.2.43 operator\*=()** [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 200 of file gslice\_array.h.

**3.56.2.44 operator\*=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 190 of file mask\_array.h.

**3.56.2.45 operator\*=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 194 of file indirect\_array.h.



**3.56.2.46 operator\*=( )** [ 4 / 6 ]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 256 of file slice\_array.h.

**3.56.2.47 operator\*=( )** [ 5 / 6 ]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
    const _Tp & __t ) [inline]
```

Multiply each element of array by *t*.

Definition at line 1106 of file valarray.

**3.56.2.48 operator\*=( )** [ 6 / 6 ]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
    const valarray< _Tp > & __v ) [inline]
```

Multiply elements of array by corresponding elements of *v*.

Definition at line 1106 of file valarray.

**3.56.2.49 operator+( )**

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+ ( )
const [inline]
```

Return a new valarray by applying unary + to each element.

Definition at line 1078 of file valarray.

**3.56.2.50 operator+=()** [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 203 of file gslice\_array.h.

**3.56.2.51 operator+=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 193 of file mask\_array.h.

**3.56.2.52 operator+=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 197 of file indirect\_array.h.

**3.56.2.53 operator+=()** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 259 of file slice\_array.h.

**3.56.2.54 operator+=()** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
    const _Tp & __t ) [inline]
```

Add *t* to each element of array.

Definition at line 1104 of file valarray.

**3.56.2.55 operator+=()** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
    const valarray< _Tp > & __v ) [inline]
```

Add corresponding elements of *v* to elements of array.

Definition at line 1104 of file valarray.

**3.56.2.56 operator-()**

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- ( ) const
[inline]
```

Return a new valarray by applying unary - to each element.

Definition at line 1079 of file valarray.

**3.56.2.57 operator-=()** [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 204 of file gslice\_array.h.

**3.56.2.58 operator-=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 194 of file mask\_array.h.

**3.56.2.59 operator-=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 198 of file indirect\_array.h.

**3.56.2.60 operator-=()** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 260 of file slice\_array.h.

**3.56.2.61 operator-=()** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
    const _Tp & __t ) [inline]
```

Subtract *t* to each element of array.

Definition at line 1105 of file valarray.

**3.56.2.62 operator-=()** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
    const valarray< _Tp > & __v ) [inline]
```

Subtract corresponding elements of *v* from elements of array.

Definition at line 1105 of file valarray.

**3.56.2.63 operator/=()** [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 201 of file gslice\_array.h.

**3.56.2.64 operator/=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 191 of file mask\_array.h.

**3.56.2.65 operator/=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 195 of file indirect\_array.h.

**3.56.2.66 operator/=()** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 257 of file slice\_array.h.

**3.56.2.67 operator/=()** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
    const _Tp & __t ) [inline]
```

Divide each element of array by *t*.

Definition at line 1107 of file valarray.

**3.56.2.68 operator/=()** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
    const valarray< _Tp > & __v ) [inline]
```

Divide elements of array by corresponding elements of *v*.

Definition at line 1107 of file valarray.

**3.56.2.69 operator<<=()** [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 208 of file gslice\_array.h.

**3.56.2.70 operator<<=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 198 of file mask\_array.h.

**3.56.2.71 operator<<=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 202 of file indirect\_array.h.

**3.56.2.72 operator<<=()** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 264 of file slice\_array.h.

**3.56.2.73 operator<<=()** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
    const _Tp & __t ) [inline]
```

Left shift each element *e* of array by *t* bits.

Definition at line 1112 of file valarray.

**3.56.2.74 operator<<=()** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) [inline]
```

Left shift elements of array by corresponding elements of *v*.

Definition at line 1112 of file valarray.

**3.56.2.75 operator=()** [1/20]

```
template<typename _Tp >
gslice_array< _Tp > & std::gslice_array< _Tp >::operator= (
    const gslice_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 148 of file gslice\_array.h.

**3.56.2.76 operator=()** [2/20]

```
template<typename _Tp >
indirect_array< _Tp > & std::indirect_array< _Tp >::operator= (
    const indirect_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file indirect\_array.h.

**3.56.2.77 operator=()** [3/20]

```
template<typename _Tp >
mask_array< _Tp > & std::mask_array< _Tp >::operator= (
    const mask_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 149 of file mask\_array.h.



**3.56.2.78 operator=()** [ 4/20 ]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator= (
    const valarray< _Tp > & __v ) const [inline]
```

Assign slice elements to corresponding elements of *v*.

Definition at line 166 of file gslice\_array.h.

References `std::valarray< _Tp >::size()`.

**3.56.2.79 operator=()** [ 5/20 ]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator= (
    const valarray< _Tp > & __v ) const [inline]
```

Assign slice elements to corresponding elements of *v*.

Definition at line 168 of file indirect\_array.h.

**3.56.2.80 operator=()** [ 6/20 ]

```
gslice & std::gslice::operator= (
    const gslice & __g ) [inline]
```

Assignment operator.

Definition at line 170 of file gslice.h.

**3.56.2.81 operator=()** [ 7/20 ]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 158 of file gslice\_array.h.

**3.56.2.82 operator=()** [8/20]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 158 of file mask\_array.h.

**3.56.2.83 operator=()** [9/20]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 163 of file indirect\_array.h.

**3.56.2.84 operator=()** [10/20]

```
template<typename _Tp >
slice_array< _Tp > & std::slice_array< _Tp >::operator= (
    const slice_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 215 of file slice\_array.h.

**3.56.2.85 operator=()** [11/20]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator= (
    const valarray< _Tp > & __v ) const [inline]
```

Assign slice elements to corresponding elements of *v*.

Definition at line 229 of file slice\_array.h.

**3.56.2.86 operator=()** [12/20]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 224 of file slice\_array.h.

**3.56.2.87 operator=()** [13/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const valarray< _Tp > & __v ) [inline]
```

Assign elements to an array.

Assign elements of array to values in *v*.

**Parameters**

<b><code>__v</code></b>	Valarray to get values from.
-------------------------	------------------------------

Definition at line 713 of file valarray.

**3.56.2.88 operator=()** [14/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    valarray< _Tp > && __v ) [inline], [noexcept]
```

Move assign elements to an array.

Move assign elements of array to values in *v*.

**Parameters**

<b><code>__v</code></b>	Valarray to get values from.
-------------------------	------------------------------

Definition at line 737 of file valarray.

**3.56.2.89 operator=()** [15/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const _Tp & __t ) [inline]
```

Assign elements to a value.

Assign all elements of array to *t*.

**Parameters**

↔	Value for elements.
↔	
↔	
↔	
<i>t</i>	

Definition at line 777 of file valarray.

**3.56.2.90 operator=()** [16/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const slice_array< _Tp > & __sa ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

**Parameters**

__sa	Array slice to get values from.
------	---------------------------------

Definition at line 785 of file valarray.

**3.56.2.91 operator=()** [17/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const gslice_array< _Tp > & __ga ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

**Parameters**

<code>__ga</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 795 of file `valarray`.

**3.56.2.92 operator=()** [18/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const mask_array< _Tp > & __ma ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

**Parameters**

<code>__ma</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 805 of file `valarray`.

**3.56.2.93 operator=()** [19/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const indirect_array< _Tp > & __ia ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

**Parameters**

<code>__↔ __ia</code>	Array slice to get values from.
---------------------------	---------------------------------

Definition at line 815 of file `valarray`.

**3.56.2.94 operator=()** [20/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    initializer_list< _Tp > __l ) [inline]
```



**3.56.2.98** `operator>>=()` [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) const [inline]
```

Right shift slice elements by corresponding elements of *v*.

Definition at line 265 of file `slice_array.h`.

**3.56.2.99** `operator>>=()` [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
    const _Tp & __t ) [inline]
```

Right shift each element *e* of array by *t* bits.

Definition at line 1113 of file `valarray`.

**3.56.2.100** `operator>>=()` [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) [inline]
```

Right shift elements of array by corresponding elements of *v*.

Definition at line 1113 of file `valarray`.

**3.56.2.101** `operator[]()` [1/9]

```
template<typename _Tp >
_Tp & std::valarray< _Tp >::operator[] (
    size_t __i ) [inline]
```

Return a reference to the *i*'th array element.

**Parameters**

↵	Index of element to return.
↵	
↵	
↵	
<i>i</i>	

**Returns**

Reference to the i'th element.

Definition at line 581 of file valarray.

**3.56.2.102 operator[]()** [2/9]

```
template<typename _Tp >
_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    slice __s ) const [inline]
```


Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

slice.

**Parameters**

<a href="#"></a> <code>__s</code>	The source slice.
--	-------------------

**Returns**

New valarray containing elements in `__s`.

Definition at line 847 of file valarray.

**3.56.2.103 operator[]()** [3/9]

```
template<typename _Tp >
slice_array< _Tp > std::valarray< _Tp >::operator[] (
    slice __s ) [inline]
```

Return a reference to an array subset.


Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

slice.



**Parameters**

<a href="#"></a> <a href="#"><u>_s</u></a>	The source slice.
--	-------------------

**Returns**

New valarray containing elements in `__s`.

Definition at line 855 of file valarray.

**3.56.2.104 operator[]()** [4/9]

```
template<typename _Tp >
_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    const gslice & __s ) const [inline]
```


Return an array subset.

Returns a slice\_array referencing the elements of the array indicated by the slice argument.

**See also**

gslice.

**Parameters**

<a href="#"></a> <a href="#"><u>_s</u></a>	The source slice.
--	-------------------

**Returns**

Slice\_array referencing elements indicated by `__s`.

Definition at line 860 of file valarray.

**3.56.2.105 operator[]()** [5/9]

```
template<typename _Tp >
gslice_array< _Tp > std::valarray< _Tp >::operator[] (
    const gslice & __s ) [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

See also

gslice.

Parameters

<code>_↔ _s</code>	The source gslice.
------------------------	--------------------

Returns

New valarray containing elements in `__s`.

Definition at line 869 of file valarray.

### 3.56.2.106 `operator[]()` [6/9]

```
template<typename _Tp >
valarray< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< bool > & __m ) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

<code>_↔ _m</code>	The valarray bitmask.
------------------------	-----------------------

Returns

New valarray containing elements indicated by `__m`.

Definition at line 877 of file valarray.


### 3.56.2.107 `operator[]()` [7/9]

```
template<typename _Tp >
mask_array< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< bool > & __m ) [inline]
```

Return a reference to an array subset.

Returns a new mask\_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

## Parameters

 <code>__m</code>	The valarray bitmask.
---	-----------------------

## Returns

New valarray containing elements indicated by `__m`.

Definition at line 889 of file valarray.





3.56.2.108 `operator[]()` [8/9]

```
template<typename _Tp >
_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    const valarray< size_t > & __i ) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

## Parameters

	The valarray element index list.
<code>__</code> 	
	
<code>__</code>  <code>i</code>	

## Returns

New valarray containing elements in `__s`.

Definition at line 900 of file valarray.

3.56.2.109 `operator[]()` [9/9]

```
template<typename _Tp >
indirect_array< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< size_t > & __i ) [inline]
```

Return a reference to an array subset.

Returns an indirect\_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect\_array refers to these elements.

## Parameters

↔	The valarray element index list.
↔	
↔	
↔	
<i>i</i>	

## Returns

Indirect\_array referencing elements in *\_\_i*.

Definition at line 908 of file valarray.

3.56.2.110 `operator^=()` [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of *v*.

Definition at line 205 of file gslice\_array.h.

3.56.2.111 `operator^=()` [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of *v*.

Definition at line 195 of file mask\_array.h.

3.56.2.112 `operator^=()` [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of *v*.

Definition at line 199 of file indirect\_array.h.

**3.56.2.113** `operator^=()` [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of *v*.

Definition at line 261 of file slice\_array.h.

**3.56.2.114** `operator^=()` [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to  $e \wedge t$ .

Definition at line 1109 of file valarray.

**3.56.2.115** `operator^=()` [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
    const valarray< _Tp > & __v ) [inline]
```

Logical xor corresponding elements of *v* with elements of array.

Definition at line 1109 of file valarray.

**3.56.2.116** `operator" |=()` [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 207 of file gslice\_array.h.

**3.56.2.117 operator" |=()** [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 197 of file mask\_array.h.

**3.56.2.118 operator" |=()** [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 201 of file indirect\_array.h.

**3.56.2.119 operator" |=()** [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 263 of file slice\_array.h.

**3.56.2.120 operator" |=()** [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to *e* | *t*.

Definition at line 1111 of file valarray.

**3.56.2.121 operator" |=()** [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
    const valarray< _Tp > & __v ) [inline]
```

Logical or corresponding elements of *v* with elements of array.

Definition at line 1111 of file valarray.

**3.56.2.122 operator~()**

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __bitwise_not >::_Rt std::valarray< _Tp >::operator~ ( )
const [inline]
```

Return a new valarray by applying unary ~ to each element.

Definition at line 1080 of file valarray.

**3.56.2.123 resize()**

```
template<class _Tp>
void std::valarray< _Tp >::resize (
    size_t __size,
    _Tp __c = _Tp() ) [inline]
```

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

**Parameters**

<code>__size</code>	New array size.
<code>__c</code>	New value for all elements.

Definition at line 1021 of file valarray.

**3.56.2.124 shift()**

```
template<class _Tp >
valarray< _Tp > std::valarray< _Tp >::shift (
    int __n ) const [inline]
```

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index  $i$ , the new position is  $i - n$ . The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements  $[0, n)$ . Negative arguments discard elements from the top of the array.

#### Parameters

<code>_↔_</code>	Number of element positions to shift.
<code>_n</code>	

#### Returns

New valarray with elements in shifted positions.

Definition at line 939 of file valarray.

#### 3.56.2.125 `size()` [1/3]

```
size_t std::slice::size ( ) const [inline]
```

Return size of slice.

Definition at line 102 of file slice\_array.h.

#### 3.56.2.126 `size()` [2/3]

```
valarray< size_t > std::gslice::size ( ) const [inline]
```

Return array of sizes of slice dimensions.

Definition at line 139 of file gslice.h.

#### 3.56.2.127 `size()` [3/3]

```
template<class _Tp >
size_t std::valarray< _Tp >::size ( ) const [inline]
```

Return the number of elements in array.

Definition at line 926 of file valarray.

Referenced by `std::gslice_array< _Tp >::operator=()`.



**3.56.2.128 start()** [1/2]

```
size_t std::slice::start ( ) const [inline]
```

Return array offset of first slice element.

Definition at line 98 of file slice\_array.h.

**3.56.2.129 start()** [2/2]

```
size_t std::gslice::start ( ) const [inline]
```

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

**3.56.2.130 stride()** [1/2]

```
size_t std::slice::stride ( ) const [inline]
```

Return array stride of slice.

Definition at line 106 of file slice\_array.h.

**3.56.2.131 stride()** [2/2]

```
valarray< size_t > std::gslice::stride ( ) const [inline]
```

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.

**3.56.2.132 sum()**

```
template<class _Tp >  
_Tp std::valarray< _Tp >::sum ( ) const [inline]
```

Return the sum of all elements in the array.

Accumulates the sum of all elements into a Tp using +=. The order of adding the elements is unspecified.

Definition at line 931 of file valarray.

**3.56.2.133 swap()**

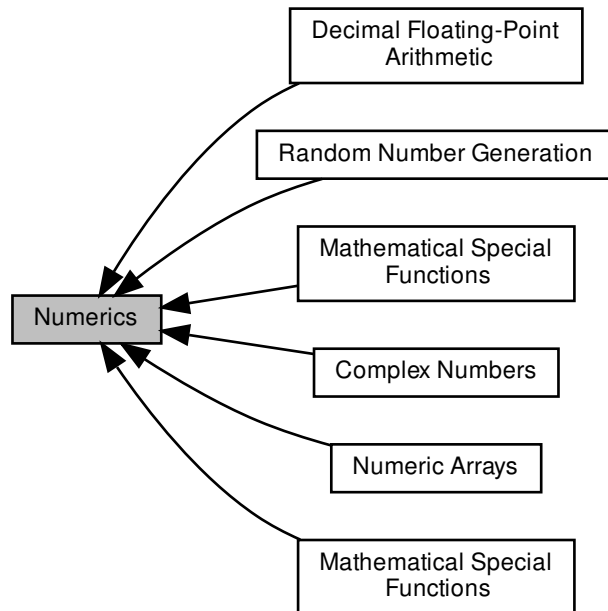
```
template<class _Tp>  
void std::valarray< _Tp >::swap (   
    valarray< _Tp > & __v ) [inline], [noexcept]
```

Swap.

Definition at line 917 of file valarray.

### 3.57 Numerics

Collaboration diagram for Numerics:



#### Modules

- [Complex Numbers](#)
- [Decimal Floating-Point Arithmetic](#)
- [Mathematical Special Functions](#)
- [Mathematical Special Functions](#)
- [Numeric Arrays](#)
- [Random Number Generation](#)

#### 3.57.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

### 3.58 Optional values

Collaboration diagram for Optional values:



#### Classes

- struct `std::experimental::fundamentals_v1::_Has_addressof< _Tp >`
- class `std::experimental::fundamentals_v1::_Optional_base< _Tp, _ShouldProvideDestructor >`
- class `std::experimental::fundamentals_v1::_Optional_base< _Tp, false >`
- class `std::experimental::fundamentals_v1::bad_optional_access`
- struct `std::experimental::fundamentals_v1::in_place_t`
- struct `std::experimental::fundamentals_v1::nullopt_t`
- class `std::experimental::fundamentals_v1::optional< _Tp >`

#### Macros

- `#define __cpp_lib_experimental_optional`

#### Typedefs

- `template<typename _Tp, typename _Up >`  
using `std::experimental::fundamentals_v1::_assigns_from_optional` = `__or_< is_assignable< _Tp &, const optional< _Up > & >, is_assignable< _Tp &, optional< _Up > & >, is_assignable< _Tp &, const optional< _Up > && >, is_assignable< _Tp &, optional< _Up > && >`
- `template<typename _Tp, typename _Up >`  
using `std::experimental::fundamentals_v1::_converts_from_optional` = `__or_< is_constructible< _Tp, const optional< _Up > & >, is_constructible< _Tp, optional< _Up > & >, is_constructible< _Tp, const optional< _Up > && >, is_constructible< _Tp, optional< _Up > && >, is_convertible< const optional< _Up > &, _Tp >, is_convertible< optional< _Up > &, _Tp >, is_convertible< const optional< _Up > &&, _Tp >, is_convertible< optional< _Up > &&, _Tp >`

## Functions

- `template<typename _Tp >`  
`constexpr enable\_if\_t<!\_Has\_addressof<\_Tp>::value, \_Tp\*> std::experimental::fundamentals_v1::__constexpr_addressof`  
`(_Tp &__t)`
- `void std::experimental::fundamentals_v1::__throw_bad_optional_access (const char *) __attribute__((__noreturn__))`
- `template<typename _Tp >`  
`constexpr optional<decay\_t<\_Tp>> std::experimental::fundamentals_v1::make_optional (_Tp &&__t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const optional<\_Tp> &__lhs, const optional<\_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const optional<\_Tp> &__lhs, nullopt\_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (nullopt\_t, const optional<\_Tp> &__rhs) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const optional<\_Tp> &__lhs, _Tp const &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const _Tp &__lhs, const optional<\_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const optional<\_Tp> &__lhs, const optional<\_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const optional<\_Tp> &, nullopt\_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (nullopt\_t, const optional<\_Tp> &__rhs) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const optional<\_Tp> &__lhs, const _Tp &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const _Tp &__lhs, const optional<\_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const optional<\_Tp> &__lhs, const optional<\_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const optional<\_Tp> &__lhs, nullopt\_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (nullopt\_t, const optional<\_Tp> &) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const optional<\_Tp> &__lhs, const _Tp &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const _Tp &__lhs, const optional<\_Tp> &__rhs)`

- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const optional< _Tp > &__lhs, const optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const optional< _Tp > &__lhs, nullopt_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (nullopt_t, const optional< _Tp > &__rhs) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const optional< _Tp > &__lhs, const _Tp &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const _Tp &__lhs, const optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const optional< _Tp > &__lhs, const optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const optional< _Tp > &__lhs, nullopt_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (nullopt_t, const optional< _Tp > &) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const optional< _Tp > &__lhs, const _Tp &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const _Tp &__lhs, const optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const optional< _Tp > &__lhs, const optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const optional< _Tp > &, nullopt_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (nullopt_t, const optional< _Tp > &__rhs) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const optional< _Tp > &__lhs, const _Tp &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const _Tp &__lhs, const optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v1::swap (optional< _Tp > &__lhs, optional< _Tp > &__rhs) noexcept(noexcept(__lhs.swap(__rhs)))`

## Variables

- `constexpr in_place_t std::experimental::fundamentals_v1::in_place`
- `constexpr nullopt_t std::experimental::fundamentals_v1::nullopt`

### 3.58.1 Detailed Description

Class template for optional values and surrounding facilities, as described in n3793 "A proposal to add a utility class to represent optional objects (Revision 5)".

### 3.58.2 Function Documentation

#### 3.58.2.1 `__constexpr_addressof()`

```
template<typename _Tp >
constexpr enable_if_t<!_Has_addressof<_Tp>::value, _Tp*> std::experimental::fundamentals_v1::__↔
constexpr_addressof (
    _Tp & __t )
```

An overload that attempts to take the address of an lvalue as a constant expression. Falls back to `__addressof` in the presence of an overloaded `addressof` operator (unary operator`&`), in which case the call will not be a constant expression.

Definition at line 175 of file `optional`.

### 3.58.3 Variable Documentation

#### 3.58.3.1 `in_place`

```
constexpr in_place_t std::experimental::fundamentals_v1::in_place
```

Tag for in-place construction.

Definition at line 88 of file `optional`.

#### 3.58.3.2 `nullopt`

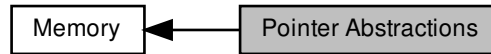
```
constexpr nullopt_t std::experimental::fundamentals_v1::nullopt
```

Tag to disengage optional objects.

Definition at line 107 of file `optional`.

### 3.59 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



#### Classes

- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- class `std::enable_shared_from_this<_Tp>`
- struct `std::hash<shared_ptr<_Tp>>`
- struct `std::hash<unique_ptr<_Tp, _Dp>>`
- struct `std::owner_less<_Tp>`
- struct `std::owner_less<shared_ptr<_Tp>>`
- struct `std::owner_less<void>`
- struct `std::owner_less<weak_ptr<_Tp>>`
- struct `std::pointer_traits<_Ptr>`
- struct `std::pointer_traits<_Tp*>`
- class `std::shared_ptr<_Tp>`
- class `std::unique_ptr<_Tp, _Dp>`
- class `std::unique_ptr<_Tp[], _Dp>`
- class `std::weak_ptr<_Tp>`

#### Macros

- `#define __cpp_lib_make_unique`

#### Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr<_Tp> std::allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename _Up>`  
`shared_ptr<_Tp> std::const_pointer_cast (const shared_ptr<_Up> &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr<_Tp> std::dynamic_pointer_cast (const shared_ptr<_Up> &__r) noexcept`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * std::get_deleter (const __shared_ptr<_Tp, _Lp> &__p) noexcept`
- `template<typename _Del, typename _Tp>`  
`_Del * std::get_deleter (const shared_ptr<_Tp> &__p) noexcept`

- `template<typename _Tp, typename... _Args>`  
`shared_ptr<_Tp> std::make_shared (_Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`_MakeUniq<_Tp>::__single_object std::make_unique (_Args &&... __args)`
- `template<typename _Tp>`  
`_MakeUniq<_Tp>::__array std::make_unique (size_t __num)`
- `template<typename _Tp, typename... _Args>`  
`_MakeUniq<_Tp>::__invalid_type std::make_unique (_Args &&...)=delete`
- `template<typename _Tp, typename _Up>`  
`bool std::operator!= (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp>`  
`bool std::operator!= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`bool std::operator!= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`bool std::operator!= (const unique_ptr<_Tp, _Dp> &__x, const unique_ptr<_Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator!= (const unique_ptr<_Tp, _Dp> &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator!= (nullptr_t, const unique_ptr<_Tp, _Dp> &__x) noexcept`
- `template<typename _Tp, typename _Up>`  
`bool std::operator< (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp>`  
`bool std::operator< (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`bool std::operator< (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`bool std::operator< (const unique_ptr<_Tp, _Dp> &__x, const unique_ptr<_Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator< (const unique_ptr<_Tp, _Dp> &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator< (nullptr_t, const unique_ptr<_Tp, _Dp> &__x)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream<_Ch, _Tr> & std::operator<< (std::basic_ostream<_Ch, _Tr> &__os, const __shared_ptr<_Tp, _Lp> &__p)`
- `template<typename _Tp, typename _Up>`  
`bool std::operator<= (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp>`  
`bool std::operator<= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`bool std::operator<= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`bool std::operator<= (const unique_ptr<_Tp, _Dp> &__x, const unique_ptr<_Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator<= (const unique_ptr<_Tp, _Dp> &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator<= (nullptr_t, const unique_ptr<_Tp, _Dp> &__x)`
- `template<typename _Tp, typename _Up>`  
`bool std::operator== (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp>`  
`bool std::operator== (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`bool std::operator== (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`



- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator== (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::operator> (const shared\_ptr< _Tp > &__a, const shared\_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator> (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator> (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator> (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator> (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator> (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Up >`  
`bool std::operator>= (const shared\_ptr< _Tp > &__a, const shared\_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator>= (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator>= (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator>= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator>= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator>= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Up >`  
`shared\_ptr< _Tp > std::static_pointer_cast (const shared\_ptr< _Up > &__r) noexcept`
- `template<typename _Tp >`  
`void std::swap (shared\_ptr< _Tp > &__a, shared\_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`  
`void std::swap (weak\_ptr< _Tp > &__a, weak\_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Dp >`  
`enable\_if< __is_swappable< _Dp >::value >::type std::swap (unique\_ptr< _Tp, _Dp > &__x, unique\_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp >`  
`bool std::atomic_is_lock_free (const shared\_ptr< _Tp > *__p)`
- `template<typename _Tp >`  
`shared\_ptr< _Tp > std::atomic_load_explicit (const shared\_ptr< _Tp > *__p, memory\_order)`
- `template<typename _Tp >`  
`shared\_ptr< _Tp > std::atomic_load (const shared\_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory\_order)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp >`  
`void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp >`  
`void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`

### 3.59.1 Detailed Description

Smart pointers, etc.

### 3.59.2 Function Documentation

#### 3.59.2.1 `allocate_shared()`

```
template<typename _Tp , typename _Alloc , typename... _Args>
shared_ptr<_Tp> std::allocate_shared (
    const _Alloc & __a,
    _Args &&... __args ) [inline]
```

Create an object that is owned by a `shared_ptr`.

##### Parameters

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

##### Returns

A `shared_ptr` that owns the newly created object.

##### Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 703 of file `bits/shared_ptr.h`.

#### 3.59.2.2 `atomic_compare_exchange_strong()` [1/2]

```
template<typename _Tp >
bool std::atomic_compare_exchange_strong (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w ) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

##### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 242 of file `shared_ptr_atomic.h`.

References `std::atomic_compare_exchange_strong_explicit()`.

**3.59.2.3 `atomic_compare_exchange_strong()`** [2/2]

```
template<typename _Tp , _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w ) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 294 of file `shared_ptr_atomic.h`.

References `std::atomic_compare_exchange_strong_explicit()`.

**3.59.2.4 `atomic_compare_exchange_strong_explicit()`** [1/2]

```
template<typename _Tp >
bool std::atomic_compare_exchange_strong_explicit (
    shared_ptr< _Tp > * __p,
```

```
shared_ptr<_Tp> * __v,  
shared_ptr<_Tp> __w,  
memory_order ,  
memory_order )
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _v	A non-null pointer to a shared_ptr object.
$\leftarrow$ _w	A non-null pointer to a shared_ptr object.

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 220 of file `shared_ptr_atomic.h`.

**3.59.2.5 atomic\_compare\_exchange\_strong\_explicit()** [2/2]

```
template<typename _Tp , _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w,
    memory_order ,
    memory_order )
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

$\leftarrow$ _p	A non-null pointer to a shared_ptr object.
$\leftarrow$ _v	A non-null pointer to a shared_ptr object.
$\leftarrow$ _w	A non-null pointer to a shared_ptr object.

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 272 of file `shared_ptr_atomic.h`.

Referenced by `std::atomic_compare_exchange_strong()`, and `std::atomic_compare_exchange_weak_explicit()`.

### 3.59.2.6 `atomic_compare_exchange_weak()` [1/2]

```
template<typename _Tp >
bool std::atomic_compare_exchange_weak (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w ) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

#### Returns

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 263 of file `shared_ptr_atomic.h`.

References `std::atomic_compare_exchange_weak_explicit()`.

### 3.59.2.7 `atomic_compare_exchange_weak()` [2/2]

```
template<typename _Tp , _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w ) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 316 of file `shared_ptr_atomic.h`.

References `std::atomic_compare_exchange_weak_explicit()`.

**3.59.2.8 `atomic_compare_exchange_weak_explicit()` [1/2]**

```
template<typename _Tp >
bool std::atomic_compare_exchange_weak_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w,
    memory_order __success,
    memory_order __failure ) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 251 of file `shared_ptr_atomic.h`.

References `std::atomic_compare_exchange_strong_explicit()`.



### 3.59.2.9 `atomic_compare_exchange_weak_explicit()` [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > * __v,
    __shared_ptr< _Tp, _Lp > __w,
    memory_order __success,
    memory_order __failure ) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

#### Returns

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 304 of file `shared_ptr_atomic.h`.

References `std::atomic_compare_exchange_strong_explicit()`.

Referenced by `std::atomic_compare_exchange_weak()`.

### 3.59.2.10 `atomic_exchange()` [1/2]

```
template<typename _Tp >
shared_ptr<_Tp> std::atomic_exchange (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r ) [inline]
```

Atomic exchange for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__r</code>	New value to store in <code>*__p</code> .

**Returns**

The original value of \*\_\_p

Definition at line 181 of file shared\_ptr\_atomic.h.



References std::atomic\_exchange\_explicit().

**3.59.2.11 atomic\_exchange()** [2/2]

```
template<typename _Tp , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::atomic_exchange (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r ) [inline]
```

Atomic exchange for shared\_ptr objects.

**Parameters**

 __p	A non-null pointer to a shared_ptr object.
 __r	New value to store in *__p.

**Returns**

The original value of \*\_\_p

Definition at line 200 of file shared\_ptr\_atomic.h.



References std::atomic\_exchange\_explicit().

**3.59.2.12 atomic\_exchange\_explicit()** [1/2]

```
template<typename _Tp >
shared_ptr<_Tp> std::atomic_exchange_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r,
    memory_order ) [inline]
```

Atomic exchange for shared\_ptr objects.

**Parameters**

 __p	A non-null pointer to a shared_ptr object.
 __r	New value to store in *__p.

**Returns**

The original value of \*\_\_p



Definition at line 171 of file shared\_ptr\_atomic.h.

**3.59.2.13 atomic\_exchange\_explicit()** [2/2]

```
template<typename _Tp , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::atomic_exchange_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r,
    memory_order ) [inline]
```

Atomic exchange for shared\_ptr objects.

**Parameters**

 __p	A non-null pointer to a shared_ptr object.
 __r	New value to store in *__p.

**Returns**

The original value of \*\_\_p

Definition at line 189 of file shared\_ptr\_atomic.h.


Referenced by std::atomic\_exchange().

**3.59.2.14 atomic\_is\_lock\_free()** [1/2]

```
template<typename _Tp , _Lock_policy _Lp>
bool std::atomic_is_lock_free (
    const __shared_ptr< _Tp, _Lp > * __p ) [inline]
```

Report whether shared\_ptr atomic operations are lock-free.

**Parameters**

 __p	A non-null pointer to a shared_ptr object.
--	--

**Returns**

True if atomic access to `*__p` is lock-free, false otherwise.

Definition at line 71 of file `shared_ptr_atomic.h`.

**3.59.2.15 `atomic_is_lock_free()` [2/2]**

```
template<typename _Tp >
bool std::atomic_is_lock_free (
    const shared_ptr< _Tp > * __p ) [inline]
```

Report whether `shared_ptr` atomic operations are lock-free.

**Parameters**

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

**Returns**

True if atomic access to `*__p` is lock-free, false otherwise.

Definition at line 82 of file `shared_ptr_atomic.h`.

**3.59.2.16 `atomic_load()` [1/2]**

```
template<typename _Tp >
shared_ptr<_Tp> std::atomic_load (
    const shared_ptr< _Tp > * __p ) [inline]
```

Atomic load for `shared_ptr` objects.

**Parameters**

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

**Returns**

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

Definition at line 106 of file `shared_ptr_atomic.h`.

References `std::atomic_load_explicit()`.

### 3.59.2.17 `atomic_load()` [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::atomic_load (
    const __shared_ptr< _Tp, _Lp > * __p ) [inline]
```

Atomic load for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

#### Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

Definition at line 119 of file `shared_ptr_atomic.h`.

References `std::atomic_load_explicit()`.

### 3.59.2.18 `atomic_load_explicit()` [1/2]

```
template<typename _Tp >
shared_ptr<_Tp> std::atomic_load_explicit (
    const shared_ptr< _Tp > * __p,
    memory_order ) [inline]
```

Atomic load for `shared_ptr` objects.

#### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

#### Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

Definition at line 98 of file `shared_ptr_atomic.h`.

#### 3.59.2.19 `atomic_load_explicit()` [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::atomic_load_explicit (
    const __shared_ptr< _Tp, _Lp > * __p,
    memory_order ) [inline]
```

Atomic load for `shared_ptr` objects.

##### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
------------------	---

##### Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

Definition at line 111 of file `shared_ptr_atomic.h`.

Referenced by `std::atomic_load()`.

#### 3.59.2.20 `atomic_store()` [1/2]

```
template<typename _Tp >
void std::atomic_store (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r ) [inline]
```

Atomic store for `shared_ptr` objects.

##### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__r</code>	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

Definition at line 143 of file `shared_ptr_atomic.h`.

References `std::atomic_store_explicit()`.

#### 3.59.2.21 `atomic_store()` [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r ) [inline]
```

Atomic store for `shared_ptr` objects.

##### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__r</code>	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

Definition at line 158 of file `shared_ptr_atomic.h`.

References `std::atomic_store_explicit()`.

#### 3.59.2.22 `atomic_store_explicit()` [1/2]

```
template<typename _Tp >
void std::atomic_store_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r,
    memory_order ) [inline]
```

Atomic store for `shared_ptr` objects.

##### Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__r</code>	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

Definition at line 134 of file `shared_ptr_atomic.h`.

**3.59.2.23 atomic\_store\_explicit()** [2/2]

```
template<typename _Tp , _Lock_policy _Lp>
void std::atomic_store_explicit (
    __shared_ptr< _Tp, _Lp > * __p,
    __shared_ptr< _Tp, _Lp > __r,
    memory_order ) [inline]
```

Atomic store for shared\_ptr objects.

**Parameters**

<code>__p</code>	A non-null pointer to a shared_ptr object.
<code>__r</code>	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

Definition at line 148 of file `shared_ptr_atomic.h`.

Referenced by `std::atomic_store()`.

**3.59.2.24 get\_deleter()**

```
template<typename _Del , typename _Tp >
_Del* std::get_deleter (
    const shared_ptr< _Tp > & __p ) [inline], [noexcept]
```

**20.7.2.2.10 shared\_ptr get\_deleter**

Definition at line 87 of file `bits/shared_ptr.h`.

**3.59.2.25 make\_shared()**

```
template<typename _Tp , typename... _Args>
shared_ptr<_Tp> std::make_shared (
    _Args &&... __args ) [inline]
```

Create an object that is owned by a shared\_ptr.

**Parameters**

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--



**Returns**

A `shared_ptr` that owns the newly created object.

**Exceptions**

<code>std::bad_alloc</code> , or	an exception thrown from the constructor of <code>_Tp</code> .
----------------------------------	--

Definition at line 718 of file `bits/shared_ptr.h`.

**3.59.2.26 make\_unique()** [1/3]

```
template<typename _Tp , typename... _Args>
_MakeUniq<_Tp>::__single_object std::make_unique (
    _Args &&... __args ) [inline]
```

`std::make_unique` for single objects

Definition at line 834 of file `unique_ptr.h`.

**3.59.2.27 make\_unique()** [2/3]

```
template<typename _Tp >
_MakeUniq<_Tp>::__array std::make_unique (
    size_t __num ) [inline]
```

`std::make_unique` for arrays of unknown bound

Definition at line 840 of file `unique_ptr.h`.

**3.59.2.28 make\_unique()** [3/3]

```
template<typename _Tp , typename... _Args>
_MakeUniq<_Tp>::__invalid_type std::make_unique (
    _Args && ... ) [inline], [delete]
```

Disable `std::make_unique` for arrays of known bound.

**3.59.2.29 operator<<()**

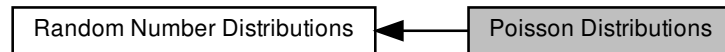
```
template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp>
std::basic_ostream<_Ch, _Tr>& std::operator<< (
    std::basic_ostream< _Ch, _Tr > & __os,
    const __shared_ptr< _Tp, _Lp > & __p ) [inline]
```

**20.7.2.2.11 shared\_ptr I/O**

Definition at line 66 of file `bits/shared_ptr.h`.

## 3.60 Poisson Distributions

Collaboration diagram for Poisson Distributions:



## Classes

- class `std::discrete_distribution< _IntType >`
- class `std::exponential_distribution< _RealType >`
- class `std::extreme_value_distribution< _RealType >`
- class `std::piecewise_constant_distribution< _RealType >`
- class `std::piecewise_linear_distribution< _RealType >`
- class `std::poisson_distribution< _IntType >`
- class `std::weibull_distribution< _RealType >`

## Functions

- template<typename \_IntType >  
bool `std::operator!=` (const `std::poisson_distribution< _IntType >` &\_\_d1, const `std::poisson_distribution< _IntType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::exponential_distribution< _RealType >` &\_\_d1, const `std::exponential_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::weibull_distribution< _RealType >` &\_\_d1, const `std::weibull_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::extreme_value_distribution< _RealType >` &\_\_d1, const `std::extreme_value_distribution< _RealType >` &\_\_d2)
- template<typename \_IntType >  
bool `std::operator!=` (const `std::discrete_distribution< _IntType >` &\_\_d1, const `std::discrete_distribution< _IntType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::piecewise_constant_distribution< _RealType >` &\_\_d1, const `std::piecewise_constant_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType >  
bool `std::operator!=` (const `std::piecewise_linear_distribution< _RealType >` &\_\_d1, const `std::piecewise_linear_distribution< _RealType >` &\_\_d2)
- template<typename \_RealType, typename \_CharT, typename \_Traits >  
`std::basic_ostream< _CharT, _Traits >` & `std::operator<<` (`std::basic_ostream< _CharT, _Traits >` &\_\_os, const `std::exponential_distribution< _RealType >` &\_\_x)

- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::extreme_value_distribution< _RealType > &__x)`

### 3.60.1 Detailed Description

### 3.60.2 Function Documentation

#### 3.60.2.1 `operator!=()` [1/7]

```
template<typename _IntType >
bool std::operator!=(
    const std::poisson_distribution< _IntType > & __d1,
    const std::poisson_distribution< _IntType > & __d2 ) [inline]
```

Return true if two Poisson distributions are different.

Definition at line 4530 of file random.h.

#### 3.60.2.2 `operator!=()` [2/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::exponential_distribution< _RealType > & __d1,
    const std::exponential_distribution< _RealType > & __d2 ) [inline]
```

Return true if two exponential distributions have different parameters.

Definition at line 4713 of file random.h.

**3.60.2.3 operator!=( )** [3/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::weibull_distribution< _RealType > & __d1,
    const std::weibull_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Weibull distributions have different parameters.

Definition at line 4921 of file random.h.

**3.60.2.4 operator!=( )** [4/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::extreme_value_distribution< _RealType > & __d1,
    const std::extreme_value_distribution< _RealType > & __d2 ) [inline]
```

Return true if two extreme value distributions have different parameters.

Definition at line 5129 of file random.h.

**3.60.2.5 operator!=( )** [5/7]

```
template<typename _IntType >
bool std::operator!=(
    const std::discrete_distribution< _IntType > & __d1,
    const std::discrete_distribution< _IntType > & __d2 ) [inline]
```

Return true if two discrete distributions have different parameters.

Definition at line 5394 of file random.h.

**3.60.2.6 operator!=( )** [6/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::piecewise_constant_distribution< _RealType > & __d1,
    const std::piecewise_constant_distribution< _RealType > & __d2 ) [inline]
```

Return true if two piecewise constant distributions have different parameters.

Definition at line 5666 of file random.h.

### 3.60.2.7 operator!=( ) [ 7 / 7 ]

```
template<typename _RealType >
bool std::operator!=(
    const std::piecewise_linear_distribution< _RealType > & __d1,
    const std::piecewise_linear_distribution< _RealType > & __d2 ) [inline]
```

Return true if two piecewise linear distributions have different parameters.

Definition at line 5940 of file random.h.

### 3.60.2.8 operator<<( ) [ 1 / 3 ]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<<(
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::exponential_distribution< _RealType > & __x )
```

Inserts a exponential\_distribution random number distribution \_\_x into the output stream \_\_os.

#### Parameters

__os	An output stream.
__x	A exponential_distribution random number distribution.

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

Definition at line 1734 of file bits/random.tcc.

References std::basic\_ios< \_CharT, \_Traits >::fill(), std::ios\_base::flags(), std::exponential\_distribution< \_RealType >::lambda(), std::left(), std::ios\_base::precision(), std::scientific(), and std::basic\_ios< \_CharT, \_Traits >::widen().

### 3.60.2.9 operator<<( ) [ 2 / 3 ]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<<(
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::weibull_distribution< _RealType > & __x )
```

Inserts a weibull\_distribution random number distribution \_\_x into the output stream \_\_os.

#### Parameters

__os	An output stream.
__x	A weibull_distribution random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2529 of file `bits/random.tcc`.

References `std::weibull_distribution< _RealType >::a()`, `std::weibull_distribution< _RealType >::b()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::ios_base::precision()`, `std::scientific()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

**3.60.2.10 operator<<() [3/3]**

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::extreme_value_distribution< _RealType > & __x )
```

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2605 of file `bits/random.tcc`.

References `std::extreme_value_distribution< _RealType >::a()`, `std::extreme_value_distribution< _RealType >::b()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::ios_base::precision()`, `std::scientific()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

**3.60.2.11 operator>>() [1/3]**

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::exponential_distribution< _RealType > & __x )
```

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A exponential_distribution random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1757 of file bits/random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution<_RealType>::param()`, and `std::skipws()`.

3.60.2.12 `operator>>()` [2/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::weibull_distribution< _RealType > & __x )
```

Extracts a weibull\_distribution random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A weibull_distribution random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2553 of file bits/random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution<_RealType>::param()`, and `std::skipws()`.

3.60.2.13 `operator>>()` [3/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::extreme_value_distribution< _RealType > & __x )
```

Extracts a extreme\_value\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>_↔ _is</code>	An input stream.
<code>_↔ _x</code>	A <code>extreme_value_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

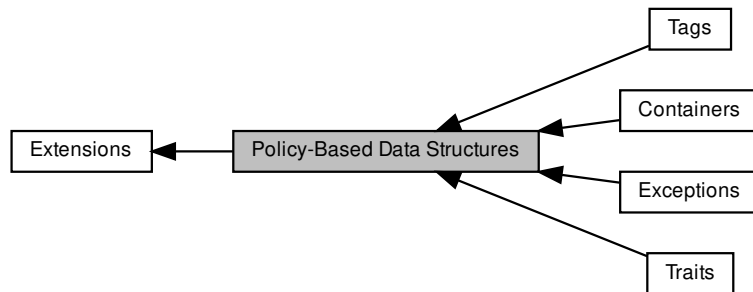
Definition at line 2629 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution<_RealType>::param()`, and `std↔  
::skipws()`.



### 3.61 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



#### Modules

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

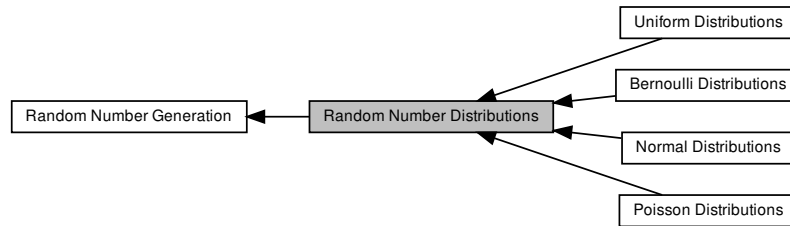
#### 3.61.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in `std` (except for some points where it differs by design).

For details, see: [http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb\\_ds/index.html](http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html)

## 3.62 Random Number Distributions

Collaboration diagram for Random Number Distributions:



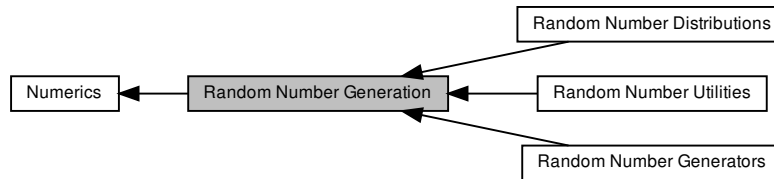
### Modules

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Uniform Distributions](#)

### 3.62.1 Detailed Description

### 3.63 Random Number Generation

Collaboration diagram for Random Number Generation:



#### Modules

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

#### Namespaces

- [std::\\_\\_detail](#)

#### Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical(_UniformRandomNumberGenerator & __g)`

#### 3.63.1 Detailed Description

A facility for generating random numbers on selected distributions.

#### 3.63.2 Function Documentation

##### 3.63.2.1 generate\_canonical()

```

template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >
_RealType std::generate_canonical (
    _UniformRandomNumberGenerator & __g )
  
```

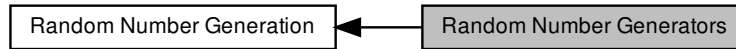
A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 3313 of file bits/random.tcc.

References `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::min()`.

## 3.64 Random Number Generators

Collaboration diagram for Random Number Generators:



## Classes

- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`
- class `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::random_device`
- class `std::shuffle_order_engine< _RandomNumberEngine, __k >`
- class `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`

## Typedefs

- typedef `minstd_rand0 std::default_random_engine`
- typedef `shuffle_order_engine< minstd_rand0, 256 > std::knuth_b`
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937`
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > std::mt19937_64`
- typedef `discard_block_engine< ranlux24_base, 223, 23 > std::ranlux24`
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > std::ranlux24_base`
- typedef `discard_block_engine< ranlux48_base, 389, 11 > std::ranlux48`
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > std::ranlux48_base`

## Functions

- template<typename \_UIntType, \_UIntType \_\_a, \_UIntType \_\_c, \_UIntType \_\_m>  
bool `std::operator!=` (const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &\_\_lhs, const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &\_\_rhs)
- template<typename \_UIntType, size\_t \_\_w, size\_t \_\_n, size\_t \_\_m, size\_t \_\_r, \_UIntType \_\_a, size\_t \_\_u, \_UIntType \_\_d, size\_t \_\_s, \_UIntType \_\_b, size\_t \_\_t, \_UIntType \_\_c, size\_t \_\_l, \_UIntType \_\_f>  
bool `std::operator!=` (const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &\_\_lhs, const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &\_\_rhs)

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const`  
`std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const`  
`std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs,`  
`const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const`  
`std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

### 3.64.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

**Table 227 Random Number Generator Requirements**

To be documented.
-------------------

### 3.64.2 Typedef Documentation

#### 3.64.2.1 `minstd_rand`

```
typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand
```

An alternative LCR (Lehmer Generator function).

Definition at line 1512 of file `random.h`.

#### 3.64.2.2 `minstd_rand0`

```
typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0
```

The classic Minimum Standard `rand0` of Lewis, Goodman, and Miller.

Definition at line 1506 of file `random.h`.

## 3.64.2.3 mt19937

```
typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL,
7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937
```

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1528 of file random.h.

## 3.64.2.4 mt19937\_64

```
typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29,
0x5555555555555555ULL, 17, 0x71d67fffed60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL> std::mt19937_64
```

An alternative Mersenne Twister.

Definition at line 1540 of file random.h.

## 3.64.3 Function Documentation

## 3.64.3.1 operator!=( ) [1/6]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool std::operator!=(
    const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
    const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs ) [inline]
```

Compares two linear congruential random number generator objects of the same type for inequality.

## Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

## Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 397 of file random.h.

### 3.64.3.2 operator!=( ) [2/6]

```
template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,
size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _U
UIntType __f>
bool std::operator!=( (
    const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __
__s, __b, __t, __c, __l, __f > & __lhs,
    const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __
__s, __b, __t, __c, __l, __f > & __rhs ) [inline]
```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for inequality.

#### Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 629 of file random.h.

### 3.64.3.3 operator!=( ) [3/6]

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
bool std::operator!=( (
    const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
    const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs ) [inline]
```

Compares two % subtract\_with\_carry\_engine random number generator objects of the same type for inequality.

#### Parameters

<code>__lhs</code>	A % subtract_with_carry_engine random number generator object.
<code>__rhs</code>	Another % subtract_with_carry_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 825 of file random.h.

3.64.3.4 `operator!=()` [4/6]

```
template<typename _RandomNumberEngine , size_t __p, size_t __r>
bool std::operator!= (
    const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
    const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs ) [inline]
```

Compares two `discard_block_engine` random number generator objects of the same type for inequality.

## Parameters

<code>__lhs</code>	A <code>discard_block_engine</code> random number generator object.
<code>__rhs</code>	Another <code>discard_block_engine</code> random number generator object.

## Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1047 of file `random.h`.

3.64.3.5 `operator!=()` [5/6]

```
template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
bool std::operator!= (
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs )
[inline]
```

Compares two `independent_bits_engine` random number generator objects of the same type for inequality.

## Parameters

<code>__lhs</code>	A <code>independent_bits_engine</code> random number generator object.
<code>__rhs</code>	Another <code>independent_bits_engine</code> random number generator object.

## Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1243 of file `random.h`.



### 3.64.3.6 operator!=( ) [ 6 / 6 ]

```
template<typename _RandomNumberEngine , size_t __k>
bool std::operator!=(
    const std::shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
    const std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs ) [inline]
```

Compares two shuffle\_order\_engine random number generator objects of the same type for inequality.

#### Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1495 of file random.h.

### 3.64.3.7 operator<<( )

```
template<typename _RandomNumberEngine , size_t __w, typename _UIntType , typename _CharT , typename
 Traits >
std::basic_ostream<_CharT, Traits>& std::operator<< (
    std::basic_ostream< _CharT, Traits > & __os,
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x )
```

Inserts the current state of a independent\_bits\_engine random number generator engine `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A independent_bits_engine random number generator engine.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1262 of file random.h.

## 3.65 Random Number Utilities

Collaboration diagram for Random Number Utilities:



### Classes

- class `std::seed_seq`

### 3.65.1 Detailed Description

### 3.66 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



#### Classes

- struct `std::ratio< _Num, _Den >`
- struct `std::ratio_equal< _R1, _R2 >`
- struct `std::ratio_not_equal< _R1, _R2 >`

#### Typedefs

- template<typename \_R1, typename \_R2 >  
using `std::ratio_divide` = typename `__ratio_divide< _R1, _R2 >::type`
- template<typename \_R1, typename \_R2 >  
using `std::ratio_multiply` = typename `__ratio_multiply< _R1, _R2 >::type`
- typedef `ratio< num, den >` `std::ratio< _Num, _Den >::type`
- typedef `ratio< __safe_multiply<(_R1::num/__gcd1),(_R2::num/__gcd2)>::value, __safe_multiply<(_R1::den/__gcd2),(_R2::den/__gcd1)>::value >` `std::ratio_multiply< _R1, _R2 >::type`
- typedef `__ratio_multiply< _R1, ratio< _R2::den, _R2::num >::type` `std::ratio_divide< _R1, _R2 >::type`

#### Variables

- static constexpr uintmax\_t `std::__big_add< __hi1, __lo1, __hi2, __lo2 >::__hi`
- static constexpr uintmax\_t `std::__big_sub< __hi1, __lo1, __hi2, __lo2 >::__hi`
- static constexpr uintmax\_t `std::__big_mul< __x, __y >::__hi`
- static constexpr uintmax\_t `std::__big_add< __hi1, __lo1, __hi2, __lo2 >::__lo`
- static constexpr uintmax\_t `std::__big_sub< __hi1, __lo1, __hi2, __lo2 >::__lo`
- static constexpr uintmax\_t `std::__big_mul< __x, __y >::__lo`
- static constexpr uintmax\_t `std::__big_div_impl< __n1, __n0, __d >::__quot`
- static constexpr uintmax\_t `std::__big_div< __n1, __n0, __d >::__quot_hi`
- static constexpr uintmax\_t `std::__big_div< __n1, __n0, __d >::__quot_lo`
- static constexpr uintmax\_t `std::__big_div_impl< __n1, __n0, __d >::__rem`
- static constexpr uintmax\_t `std::__big_div< __n1, __n0, __d >::__rem`
- static constexpr intmax\_t `std::ratio< _Num, _Den >::den`
- static constexpr intmax\_t `std::__ratio_multiply< _R1, _R2 >::den`
- static constexpr intmax\_t `std::__ratio_divide< _R1, _R2 >::den`
- static constexpr intmax\_t `std::ratio< _Num, _Den >::num`
- static constexpr intmax\_t `std::__ratio_multiply< _R1, _R2 >::num`
- static constexpr intmax\_t `std::__ratio_divide< _R1, _R2 >::num`
- static const intmax\_t `std::__safe_multiply< _Pn, _Qn >::value`

### 3.66.1 Detailed Description

Compile time representation of finite rational numbers.

### 3.66.2 Typedef Documentation

#### 3.66.2.1 ratio\_divide

```
template<typename _R1 , typename _R2 >  
using std::ratio_divide = typedef typename __ratio_divide<_R1, _R2>::type
```

ratio\_divide

Definition at line 336 of file ratio.

#### 3.66.2.2 ratio\_multiply

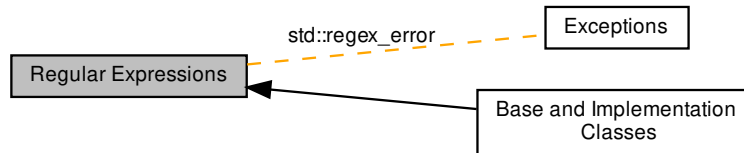
```
template<typename _R1 , typename _R2 >  
using std::ratio_multiply = typedef typename __ratio_multiply<_R1, _R2>::type
```

ratio\_multiply

Definition at line 313 of file ratio.

### 3.67 Regular Expressions

Collaboration diagram for Regular Expressions:



#### Modules

- [Base and Implementation Classes](#)

#### Namespaces

- [std::regex\\_constants](#)

#### Classes

- class [std::basic\\_regex< \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::match\\_results< \\_Bi\\_iter, \\_Alloc >](#)
- class [std::regex\\_error](#)
- class [std::regex\\_iterator< \\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::regex\\_token\\_iterator< \\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::regex\\_traits< \\_Ch\\_type >](#)
- class [std::sub\\_match< \\_Biter >](#)

#### Typedefs

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`using std::__sub_match_string = basic\_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `typedef match\_results< const char * > std::cmatch`
- `typedef regex\_iterator< const char * > std::cregex_iterator`
- `typedef regex\_token\_iterator< const char * > std::cregex\_token\_iterator`
- `typedef sub\_match< const char * > std::csub\_match`
- `typedef basic\_regex< char > std::regex`
- `typedef match\_results< string::const_iterator > std::smatch`
- `typedef regex\_iterator< string::const_iterator > std::sregex_iterator`
- `typedef regex\_token\_iterator< string::const_iterator > std::sregex\_token\_iterator`

- typedef `sub_match`< string::const\_iterator > `std::ssub_match`
- typedef `match_results`< const wchar\_t \* > `std::wcmatch`
- typedef `regex_iterator`< const wchar\_t \* > `std::wcregex_iterator`
- typedef `regex_token_iterator`< const wchar\_t \* > `std::wcregex_token_iterator`
- typedef `sub_match`< const wchar\_t \* > `std::wcsub_match`
- typedef `basic_regex`< wchar\_t > `std::wregex`
- typedef `match_results`< wstring::const\_iterator > `std::wsmatch`
- typedef `regex_iterator`< wstring::const\_iterator > `std::wsregex_iterator`
- typedef `regex_token_iterator`< wstring::const\_iterator > `std::wsregex_token_iterator`
- typedef `sub_match`< wstring::const\_iterator > `std::wssub_match`

## Functions

- template<typename `_Bilter` >  
bool `std::operator!=` (const `sub_match`< `_Bilter` > &\_\_lhs, const `sub_match`< `_Bilter` > &\_\_rhs)
- template<typename `_Bi_iter`, typename `_Ch_traits`, typename `_Ch_alloc` >  
bool `std::operator!=` (const `__sub_match_string`< `_Bi_iter`, `_Ch_traits`, `_Ch_alloc` > &\_\_lhs, const `sub_match`< `_Bi_iter` > &\_\_rhs)
- template<typename `_Bi_iter`, typename `_Ch_traits`, typename `_Ch_alloc` >  
bool `std::operator!=` (const `sub_match`< `_Bi_iter` > &\_\_lhs, const `__sub_match_string`< `_Bi_iter`, `_Ch_traits`, `_Ch_alloc` > &\_\_rhs)
- template<typename `_Bi_iter` >  
bool `std::operator!=` (typename iterator\_traits< `_Bi_iter` >::value\_type const \*\_\_lhs, const `sub_match`< `_Bi_iter` > &\_\_rhs)
- template<typename `_Bi_iter` >  
bool `std::operator!=` (const `sub_match`< `_Bi_iter` > &\_\_lhs, typename iterator\_traits< `_Bi_iter` >::value\_type const \*\_\_rhs)
- template<typename `_Bi_iter` >  
bool `std::operator!=` (typename iterator\_traits< `_Bi_iter` >::value\_type const &\_\_lhs, const `sub_match`< `_Bi_iter` > &\_\_rhs)
- template<typename `_Bi_iter` >  
bool `std::operator!=` (const `sub_match`< `_Bi_iter` > &\_\_lhs, typename iterator\_traits< `_Bi_iter` >::value\_type const &\_\_rhs)
- template<typename `_Bi_iter`, class `_Alloc` >  
bool `std::operator!=` (const `match_results`< `_Bi_iter`, `_Alloc` > &\_\_m1, const `match_results`< `_Bi_iter`, `_Alloc` > &\_\_m2)
- template<typename `_Bilter` >  
bool `std::operator<` (const `sub_match`< `_Bilter` > &\_\_lhs, const `sub_match`< `_Bilter` > &\_\_rhs)
- template<typename `_Bi_iter`, typename `_Ch_traits`, typename `_Ch_alloc` >  
bool `std::operator<` (const `__sub_match_string`< `_Bi_iter`, `_Ch_traits`, `_Ch_alloc` > &\_\_lhs, const `sub_match`< `_Bi_iter` > &\_\_rhs)
- template<typename `_Bi_iter`, class `_Ch_traits`, class `_Ch_alloc` >  
bool `std::operator<` (const `sub_match`< `_Bi_iter` > &\_\_lhs, const `__sub_match_string`< `_Bi_iter`, `_Ch_traits`, `_Ch_alloc` > &\_\_rhs)
- template<typename `_Bi_iter` >  
bool `std::operator<` (typename iterator\_traits< `_Bi_iter` >::value\_type const \*\_\_lhs, const `sub_match`< `_Bi_iter` > &\_\_rhs)
- template<typename `_Bi_iter` >  
bool `std::operator<` (const `sub_match`< `_Bi_iter` > &\_\_lhs, typename iterator\_traits< `_Bi_iter` >::value\_type const \*\_\_rhs)

- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`  
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bilter >`  
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`  
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`

- `template<typename _Bilter >`  
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`  
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`



- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`  
`bool std::regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool std::regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`  
`bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _Fst, _Fsa > & __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, const`

```

basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __↵
flags=regex_constants::match_default)
• template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >
basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx↵
_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↵
flags=regex_constants::match_default)
• template<typename _Rx_traits, typename _Ch_type >
basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits
> &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

```

## Constants

std [28.8.1](1)

- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::icase
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::nosubs
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::optimize
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::collate
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::ECMAScript
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::basic
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::extended
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::awk
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::grep
- static constexpr flag\_type std::basic\_regex< \_Ch\_type, \_Rx\_traits >::egrep

### 3.67.1 Detailed Description

A facility for performing regular expression pattern matching.

### 3.67.2 Typedef Documentation

#### 3.67.2.1 cregex\_token\_iterator

```
typedef regex_token_iterator<const char*> std::cregex_token_iterator
```

Token iterator for C-style NULL-terminated strings.

Definition at line 2820 of file regex.h.

### 3.67.2.2 csub\_match

```
typedef sub_match<const char*> std::csub_match
```

Standard regex submatch over a C-style null-terminated string.

Definition at line 957 of file regex.h.

### 3.67.2.3 regex

```
typedef basic_regex<char> std::regex
```

Standard regular expressions.

Definition at line 829 of file regex.h.

### 3.67.2.4 sregex\_token\_iterator

```
typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator
```

Token iterator for standard strings.

Definition at line 2823 of file regex.h.

### 3.67.2.5 ssub\_match

```
typedef sub_match<string::const_iterator> std::ssub_match
```

Standard regex submatch over a standard string.

Definition at line 960 of file regex.h.

### 3.67.2.6 wcregex\_token\_iterator

```
typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator
```

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2827 of file regex.h.

### 3.67.2.7 wsub\_match

```
typedef sub_match<const wchar_t*> std::wsub_match
```

Regex submatch over a C-style null-terminated wide string.

Definition at line 964 of file regex.h.

### 3.67.2.8 wregex

```
typedef basic_regex<wchar_t> std::wregex
```

Standard wide-character regular expressions.

Definition at line 833 of file regex.h.

### 3.67.2.9 wsregex\_token\_iterator

```
typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator
```

Token iterator for standard wide-character strings.

Definition at line 2830 of file regex.h.

### 3.67.2.10 wssub\_match

```
typedef sub_match<wstring::const_iterator> std::wssub_match
```

Regex submatch over a standard wide string.

Definition at line 967 of file regex.h.

## 3.67.3 Function Documentation

### 3.67.3.1 operator!=( ) [1/8]

```
template<typename _BiIter >  
bool std::operator!=(  
    const sub_match<_BiIter > & __lhs,  
    const sub_match<_BiIter > & __rhs ) [inline]
```

Tests the inequivalence of two regular expression submatches.

**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 991 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**3.67.3.2 operator"!="() [2/8]**

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator!= (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1069 of file `regex.h`.

**3.67.3.3 operator"!="() [3/8]**

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator!= (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1149 of file `regex.h`.

3.67.3.4 `operator!=( )` [4/8]

```
template<typename _Bi_iter >
bool std::operator!=(
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub\_match< _Bi_iter > & __rhs ) [inline]
```

Tests the inequivalence of an iterator value and a regular expression submatch.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1226 of file `regex.h`.

3.67.3.5 `operator!=( )` [5/8]

```
template<typename _Bi_iter >
bool std::operator!=(
    const sub\_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string.

**Returns**

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1300 of file `regex.h`.

**3.67.3.6 `operator!=( )`** [6/8]

```
template<typename _Bi_iter >
bool std::operator!= (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1377 of file `regex.h`.

**3.67.3.7 `operator!=( )`** [7/8]

```
template<typename _Bi_iter >
bool std::operator!= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

**Returns**

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1457 of file regex.h.

### 3.67.3.8 operator!=( ) [ 8 / 8 ]

```
template<typename _Bi_iter , class _Alloc >
bool std::operator!=(
    const match_results< _Bi_iter, _Alloc > & __m1,
    const match_results< _Bi_iter, _Alloc > & __m2 ) [inline]
```

Compares two match\_results for inequality.

#### Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1981 of file regex.h.

### 3.67.3.9 operator<( ) [ 1 / 7 ]

```
template<typename _BiIter >
bool std::operator<(
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [inline]
```

Tests the ordering of two regular expression submatches.

#### Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

#### Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1002 of file regex.h.

References `std::sub_match< _BiIter >::compare()`.



**3.67.3.10** `operator<()` [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator< (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1081 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

3.67.3.11 `operator<()` [3/7]

```
template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator< (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1161 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

3.67.3.12 `operator<()` [4/7]

```
template<typename _Bi_iter >
bool std::operator< (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1238 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**3.67.3.13 operator<() [5/7]**

```
template<typename _Bi_iter >
bool std::operator< (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1312 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**3.67.3.14 operator<() [6/7]**

```
template<typename _Bi_iter >
bool std::operator< (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1389 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

3.67.3.15 `operator<()` [7/7]

```
template<typename _Bi_iter >
bool std::operator< (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

## Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1469 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

3.67.3.16 `operator<<()`

```
template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter >
basic_ostream<_Ch_type, _Ch_traits>& std::operator<< (
    basic_ostream< _Ch_type, _Ch_traits > & __os,
    const sub_match< _Bi_iter > & __m ) [inline]
```

Inserts a matched string into an output stream.

**Parameters**

<code>__os</code>	The output stream.
<code>__m</code>	A submatch string.

**Returns**

the output stream with the submatch string inserted.

Definition at line 1523 of file `regex.h`.

References `std::sub_match<_Bilter>::str()`.

**3.67.3.17 operator<=()** [1/7]

```
template<typename _Bilter >
bool std::operator<= (
    const sub_match<_Bilter> & __lhs,
    const sub_match<_Bilter> & __rhs ) [inline]
```

Tests the ordering of two regular expression submatches.

**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1013 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

**3.67.3.18 operator<=()** [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator<= (
    const sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc> & __lhs,
    const sub_match<_Bi_iter> & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1120 of file `regex.h`.

3.67.3.19 `operator<=()` [3/7]

```
template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator<= (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1200 of file `regex.h`.

3.67.3.20 `operator<=()` [4/7]

```
template<typename _Bi_iter >
bool std::operator<= (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1274 of file `regex.h`.

**3.67.3.21 operator<=()** [5/7]

```
template<typename _Bi_iter >
bool std::operator<= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1348 of file `regex.h`.

**3.67.3.22 operator<=()** [6/7]

```
template<typename _Bi_iter >
bool std::operator<= (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1428 of file regex.h.

### 3.67.3.23 operator<=() [7/7]

```
template<typename _Bi_iter >
bool std::operator<= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

#### Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

#### Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1508 of file regex.h.

### 3.67.3.24 operator==( [1/8]

```
template<typename _BiIter >
bool std::operator== (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [inline]
```

Tests the equivalence of two regular expression submatches.

#### Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

#### Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 980 of file regex.h.

References `std::sub_match< _BiIter >::compare()`.



**3.67.3.25 operator==( )** [2/8]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator==(
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the equivalence of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1053 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**3.67.3.26 operator==( )** [3/8]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator==(
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the equivalence of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1133 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**3.67.3.27** `operator==( )` [4/8]

```
template<typename _Bi_iter >
bool std::operator==(
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the equivalence of a C string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A C string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1213 of file `regex.h`.

References `std::sub_match< _Biliter >::compare()`.

**3.67.3.28** `operator==( )` [5/8]

```
template<typename _Bi_iter >
bool std::operator==(
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the equivalence of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string?

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1287 of file `regex.h`.

References `std::sub_match< _Biliter >::compare()`.

**3.67.3.29 operator==( )** [6/8]

```
template<typename _Bi_iter >
bool std::operator==(
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the equivalence of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1361 of file regex.h.

References `std::sub_match< _Bilter >::compare()`.

**3.67.3.30 operator==( )** [7/8]

```
template<typename _Bi_iter >
bool std::operator==(
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the equivalence of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1441 of file regex.h.

References `std::sub_match< _Bilter >::compare()`.

**3.67.3.31 operator==( )** [8/8]

```
template<typename _Bi_iter , typename _Alloc >
bool std::operator== (
    const match_results< _Bi_iter, _Alloc > & __m1,
    const match_results< _Bi_iter, _Alloc > & __m2 ) [inline]
```

Compares two match\_results for equality.

**Returns**

true if the two objects refer to the same match, false otherwise.

Definition at line 1957 of file regex.h.

References std::equal().

**3.67.3.32 operator>()** [1/7]

```
template<typename _BiIter >
bool std::operator> (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [inline]
```

Tests the ordering of two regular expression submatches.

**Parameters**

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

**Returns**

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1035 of file regex.h.

References std::sub\_match<\_BiIter>::compare().

**3.67.3.33 operator>()** [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator> (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1096 of file `regex.h`.

**3.67.3.34 `operator>()`** [3/7]

```
template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator> (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1176 of file `regex.h`.

**3.67.3.35 `operator>()`** [4/7]

```
template<typename _Bi_iter >
bool std::operator> (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1250 of file `regex.h`.

**3.67.3.36 operator>() [5/7]**

```
template<typename _Bi_iter >
bool std::operator> (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1324 of file `regex.h`.

**3.67.3.37 operator>() [6/7]**

```
template<typename _Bi_iter >
bool std::operator> (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1404 of file regex.h.

### 3.67.3.38 operator>() [7/7]

```
template<typename _Bi_iter >
bool std::operator> (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

#### Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

#### Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1484 of file regex.h.

### 3.67.3.39 operator>=() [1/7]

```
template<typename _BiIter >
bool std::operator>= (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [inline]
```

Tests the ordering of two regular expression submatches.

#### Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

#### Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1024 of file regex.h.

References `std::sub_match< _BiIter >::compare()`.

**3.67.3.40** `operator>=()` [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool std::operator>= (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

**Parameters**

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1108 of file `regex.h`.

**3.67.3.41** `operator>=()` [3/7]

```
template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator>= (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1188 of file `regex.h`.

**3.67.3.42** `operator>=()` [4/7]

```
template<typename _Bi_iter >
bool std::operator>= (
```



```
typename iterator_traits< _Bi_iter >::value_type const * __lhs,  
const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

## Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1262 of file `regex.h`.

**3.67.3.43** `operator>=()` [5/7]

```
template<typename _Bi_iter >
bool std::operator>= (
    const sub\_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

## Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

## Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1336 of file `regex.h`.

**3.67.3.44** `operator>=()` [6/7]

```
template<typename _Bi_iter >
bool std::operator>= (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub\_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

## Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1416 of file `regex.h`.

**3.67.3.45 operator>=()** [7/7]

```
template<typename _Bi_iter >
bool std::operator>= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

**Parameters**

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

**Returns**

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1496 of file `regex.h`.

**3.67.3.46 regex\_match()** [1/7]

```
template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
bool std::regex_match (
    _Bi_iter __s,
    _Bi_iter __e,
    match_results< _Bi_iter, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

**Parameters**

<code>__s</code>	Start of the character sequence to match.
<code>__e</code>	One-past-the-end of the character sequence to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

## Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2025 of file `regex.h`.

Referenced by `std::regex_match()`.

**3.67.3.47** `regex_match()` [2/7]

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
bool std::regex_match (
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

## Parameters

<i>__first</i>	Beginning of the character sequence to match.
<i>__last</i>	One-past-the-end of the character sequence to match.
<i>__re</i>	The regular expression.
<i>__flags</i>	Controls how the regular expression is matched.

## Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2053 of file `regex.h`.

References `std::regex_match()`.

**3.67.3.48 regex\_match()** [3/7]

```
template<typename _Ch_type , typename _Alloc , typename _Rx_traits >
bool std::regex_match (
    const _Ch_type * __s,
    match_results< const _Ch_type *, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression *e* and a C-style null-terminated string.

**Parameters**

<code>__s</code>	The C-style null-terminated string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

**Return values**

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

**Exceptions**

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2078 of file `regex.h`.

References `std::regex_match()`.

**3.67.3.49 regex\_match()** [4/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
_Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression *e* and a string.

## Parameters

<code>__s</code>	The string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2102 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_match()`.

3.67.3.50 `regex_match()` [5/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & ,
    const basic_regex< _Ch_type, _Rx_traits > & ,
    regex_constants::match_flag_type = regex_constants::match_default ) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

3.67.3.51 `regex_match()` [6/7]

```
template<typename _Ch_type , class _Rx_traits >
bool std::regex_match (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

## Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2138 of file `regex.h`.

References `std::regex_match()`.

3.67.3.52 `regex_match()` [7/7]

```
template<typename _Ch_traits , typename _Str_allocator , typename _Ch_type , typename _Rx_traits
>
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and a string.

## Parameters

<code>__s</code>	[IN] The string to match.
<code>__re</code>	[IN] The regular expression.
<code>__flags</code>	[IN] Controls how the regular expression is matched.

## Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2160 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_match()`.

3.67.3.53 `regex_replace()` [1/6]

```
template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type , typename
_St , typename _Sa >
_Out_iter std::regex_replace (
    _Out_iter __out,
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const basic_string<_Ch_type, _St, _Sa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

## Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

`__out`

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2331 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`.

Referenced by `std::regex_replace()`.



**3.67.3.54** `regex_replace()` [2/6]

```
template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type >
_Out_iter std::regex_replace (
    _Out_iter __out,
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default )
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

**Parameters**

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

**Returns**

`__out`

**Exceptions**

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 465 of file `regex.tcc`.

References `std::pair<_T1, _T2>::first`, `std::regex_constants::format_first_only`, `std::regex_constants::format_no_copy`, and `std::pair<_T1, _T2>::second`.

**3.67.3.55** `regex_replace()` [3/6]

```
template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa , typename _Fst ,
typename _Fsa >
basic_string<_Ch_type, _St, _Sa> std::regex_replace (
    const basic_string< _Ch_type, _St, _Sa > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
```

```
const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,  
regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

## Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

The string after replacing.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2376 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

3.67.3.56 `regex_replace()` [4/6]

```
template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa >
basic_string<_Ch_type, _St, _Sa> std::regex_replace (
    const basic_string<_Ch_type, _St, _Sa > & __s,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

## Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

The string after replacing.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2402 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

3.67.3.57 `regex_replace()` [5/6]

```
template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa >
basic_string<_Ch_type> std::regex_replace (
    const _Ch_type * __s,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const basic_string<_Ch_type, _St, _Sa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

## Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

## Returns

The string after replacing.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2428 of file `regex.h`.

References `std::back_inserter()`, and `std::regex_replace()`.

3.67.3.58 `regex_replace()` [6/6]

```
template<typename _Rx_traits , typename _Ch_type >
basic_string<_Ch_type> std::regex_replace (
```

```

const _Ch_type * __s,
const basic_regex< _Ch_type, _Rx_traits > & __e,
const _Ch_type * __fmt,
regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]

```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

#### Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

#### Returns

The string after replacing.

#### Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2454 of file `regex.h`.

References `std::back_inserter()`, and `std::regex_replace()`.

#### 3.67.3.59 `regex_search()` [1/7]

```

template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    _Bi_iter __s,
    _Bi_iter __e,
    match_results< _Bi_iter, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]

```

Searches for a regular expression within a range.

#### Parameters

<code>__s</code>	[IN] The start of the string to search.
<code>__e</code>	[IN] One-past-the-end of the string to search.
<code>__m</code>	[OUT] The match results.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

## Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <code>m</code> is undefined.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2183 of file `regex.h`.

Referenced by `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator++()`, `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >::regex_iterator()`, and `std::regex_search()`.

3.67.3.60 `regex_search()` [2/7]

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex<_Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

## Parameters

<i>__first</i>	[IN] The start of the string to search.
<i>__last</i>	[IN] One-past-the-end of the string to search.
<i>__re</i>	[IN] The regular expression to search for.
<i>__flags</i>	[IN] Search policy flags.

## Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2207 of file `regex.h`.

References `std::regex_search()`.

### 3.67.3.61 `regex_search()` [3/7]

```
template<typename _Ch_type , class _Alloc , class _Rx_traits >
bool std::regex_search (
    const _Ch_type * __s,
    match_results< const _Ch_type *, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

#### Parameters

<code>__s</code>	[IN] A C-string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

#### Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

#### Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2230 of file `regex.h`.

References `std::regex_search()`.

### 3.67.3.62 `regex_search()` [4/7]

```
template<typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

## Parameters

<code>_↔ _s</code>	[IN] The C-string to search.
<code>_↔ _e</code>	[IN] The regular expression to search for.
<code>_↔ _f</code>	[IN] Search policy flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

## Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2249 of file `regex.h`.

References `std::regex_search()`.

3.67.3.63 `regex_search()` [5/7]

```
template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_↔
traits >
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

## Parameters

<code>__s</code>	[IN] The string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.



## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2268 of file `regex.h`.

References `std::regex_search()`.

3.67.3.64 `regex_search()` [6/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
    _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

## Parameters

<code>__s</code>	[IN] A C++ string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

## Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

## Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2291 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

**3.67.3.65** `regex_search()` [7/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & ,
    const basic_regex< _Ch_type, _Rx_traits > & ,
    regex_constants::match_flag_type = regex_constants::match_default ) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

**3.67.3.66** `swap()` [1/2]

```
template<typename _Ch_type , typename _Rx_traits >
void std::swap (
    basic_regex< _Ch_type, _Rx_traits > & __lhs,
    basic_regex< _Ch_type, _Rx_traits > & __rhs ) [inline]
```

Swaps the contents of two regular expression objects.

**Parameters**

<code>__lhs</code>	First regular expression.
<code>__rhs</code>	Second regular expression.

Definition at line 845 of file `regex.h`.

**3.67.3.67** `swap()` [2/2]

```
template<typename _Bi_iter , typename _Alloc >
void std::swap (
    match_results< _Bi_iter, _Alloc > & __lhs,
    match_results< _Bi_iter, _Alloc > & __rhs ) [inline]
```

Swaps two match results.

**Parameters**

<code>__lhs</code>	A match result.
<code>__rhs</code>	A match result.

The contents of the two `match_results` objects are swapped.

Definition at line 1995 of file `regex.h`.

### 3.68 SGI

Collaboration diagram for SGI:



#### Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
- struct `__gnu_cxx::constant_void_fun< _Result >`
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
- struct `__gnu_cxx::select2nd< _Pair >`
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`

#### Functions

- template<typename `_Tp` >  
const `_Tp` & `__gnu_cxx::__median` (const `_Tp` & `__a`, const `_Tp` & `__b`, const `_Tp` & `__c`)
- template<typename `_Tp`, typename `_Compare` >  
const `_Tp` & `__gnu_cxx::__median` (const `_Tp` & `__a`, const `_Tp` & `__b`, const `_Tp` & `__c`, `_Compare` `__comp`)
- size\_t `std::bitset< _Nb >::Find_first` () const noexcept
- size\_t `std::bitset< _Nb >::Find_next` (size\_t `__prev`) const noexcept
- template<class `_Operation1`, class `_Operation2` >  
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1` (const `_Operation1` & `__fn1`, const `_Operation2` & `__fn2`)
- template<class `_Operation1`, class `_Operation2`, class `_Operation3` >  
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2` (const `_Operation1` & `__fn1`, const `_Operation2` & `__fn2`, const `_Operation3` & `__fn3`)

- `template<class _Result >`  
`constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)`
  - `template<class _Result >`  
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result &__val)`
  - `template<class _Result >`  
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)`
  - `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
  - `template<typename _InputIterator, typename _Distance >`  
`void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
  - `template<class _Tp >`  
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
  - `template<class _Tp >`  
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
  - `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
  - `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
  - `template<typename _Tp, typename _Integer >`  
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
  - `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
  - `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)`
  - `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`
- 
- `bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos) noexcept`
  - `bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos, int __val) noexcept`
  - `bitset< _Nb > & std::bitset< _Nb >::Unchecked_reset (size_t __pos) noexcept`
  - `bitset< _Nb > & std::bitset< _Nb >::Unchecked_flip (size_t __pos) noexcept`
  - `constexpr bool std::bitset< _Nb >::Unchecked_test (size_t __pos) const noexcept`

### 3.68.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

**NB** that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function `compose2` takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

## 3.68.2 Function Documentation

### 3.68.2.1 `__median()` [1/2]

```
template<typename _Tp >
const _Tp& __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c )
```

Find the median of three values.

**Parameters**

$\_a$	A value.
$\_b$	A value.
$\_c$	A value.

**Returns**

One of  $a$ ,  $b$  or  $c$ .

If  $\{1,m,n\}$  is some convolution of  $\{a,b,c\}$  such that  $1 \leq m \leq n$  then the value returned will be  $m$ . This is an SGI extension.

Definition at line 546 of file `ext/algorithm`.

**3.68.2.2 `__median()`** [2/2]

```
template<typename _Tp , typename _Compare >
const _Tp& __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c,
    _Compare __comp )
```

Find the median of three values using a predicate for comparison.

**Parameters**

$\_\_a$	A value.
$\_\_b$	A value.
$\_\_c$	A value.
$\_\_comp$	A binary predicate.

**Returns**

One of  $a$ ,  $b$  or  $c$ .

If  $\{1,m,n\}$  is some convolution of  $\{a,b,c\}$  such that `comp(1, m)` and `comp(m, n)` are both true then the value returned will be  $m$ . This is an SGI extension.

Definition at line 580 of file `ext/algorithm`.

### 3.68.2.3 `_Find_first()`

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_first ( ) const [inline], [noexcept]
```

Finds the index of the first "on" bit.

#### Returns

The index of the first bit set, or `size()` if not found.

#### See also

[`\_Find\_next`](#)

Definition at line 1367 of file `bitset`.

### 3.68.2.4 `_Find_next()`

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_next (
    size_t __prev ) const [inline], [noexcept]
```

Finds the index of the next "on" bit after `prev`.

#### Returns

The index of the next bit set, or `size()` if not found.

#### Parameters

<code>__prev</code>	Where to start searching.
---------------------	---------------------------

#### See also

[`\_Find\_first`](#)

Definition at line 1378 of file `bitset`.

### 3.68.2.5 `_Unchecked_flip()`

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_flip (
    size_t __pos ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1054 of file `bitset`.

#### 3.68.2.6 `_Unchecked_reset()`

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_reset (
    size_t __pos ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1047 of file `bitset`.

#### 3.68.2.7 `_Unchecked_set()` [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_set (
    size_t __pos ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1030 of file `bitset`.

#### 3.68.2.8 `_Unchecked_set()` [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_set (
    size_t __pos,
    int __val ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1037 of file `bitset`.

#### 3.68.2.9 `_Unchecked_test()`

```
template<size_t _Nb>
constexpr bool std::bitset<_Nb>::_Unchecked_test (
    size_t __pos ) const [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1061 of file `bitset`.



#### 3.68.2.10 compose1()

```
template<class _Operation1 , class _Operation2 >
unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2 ) [inline]
```

An [SGI extension](#) .

Definition at line 145 of file ext/functional.

#### 3.68.2.11 compose2()

```
template<class _Operation1 , class _Operation2 , class _Operation3 >
binary_compose<_Operation1, _Operation2, _Operation3> __gnu_cxx::compose2 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2,
    const _Operation3 & __fn3 ) [inline]
```

An [SGI extension](#) .

Definition at line 172 of file ext/functional.

#### 3.68.2.12 constant0()

```
template<class _Result >
constant_void_fun<_Result> __gnu_cxx::constant0 (
    const _Result & __val ) [inline]
```

An [SGI extension](#) .

Definition at line 330 of file ext/functional.

#### 3.68.2.13 constant1()

```
template<class _Result >
constant_unary_fun<_Result, _Result> __gnu_cxx::constant1 (
    const _Result & __val ) [inline]
```

An [SGI extension](#) .

Definition at line 336 of file ext/functional.

3.68.2.14 `constant2()`

```
template<class _Result >
constant_binary_fun<_Result,_Result,_Result> __gnu_cxx::constant2 (
    const _Result & __val ) [inline]
```

An [SGI extension](#) .

Definition at line 342 of file `ext/functional`.

3.68.2.15 `copy_n()`

```
template<typename _InputIterator , typename _Size , typename _OutputIterator >
pair<_InputIterator, _OutputIterator> __gnu_cxx::copy_n (
    _InputIterator __first,
    _Size __count,
    _OutputIterator __result ) [inline]
```

Copies the range `[first,first+count)` into `[result,result+count)`.

## Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

## Returns

A `std::pair` composed of `first+count` and `result+count`.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 120 of file `ext/algorithm`.

3.68.2.16 `distance()`

```
template<typename _InputIterator , typename _Distance >
void __gnu_cxx::distance (
    _InputIterator __first,
    _InputIterator __last,
    _Distance & __n ) [inline]
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 105 of file `ext/iterator`.

**3.68.2.17 identity\_element()** [1/2]

```
template<class _Tp >
_Tp __gnu_cxx::identity_element (
    std::plus< _Tp > ) [inline]
```

An [SGI extension](#) .

Definition at line 87 of file ext/functional.

**3.68.2.18 identity\_element()** [2/2]

```
template<class _Tp >
_Tp __gnu_cxx::identity_element (
    std::multiplies< _Tp > ) [inline]
```

An [SGI extension](#) .

Definition at line 93 of file ext/functional.

**3.68.2.19 lexicographical\_compare\_3way()**

```
template<typename _InputIterator1 , typename _InputIterator2 >
int __gnu_cxx::lexicographical_compare_3way (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2 )
```

memcmp on steroids.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

**Returns**

An int, as with memcmp.

The return value will be less than zero if the first range is *lexicographically less than* the second, greater than zero if the second range is *lexicographically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 201 of file ext/algorithm.

**3.68.2.20 power()** [1/2]

```
template<typename _Tp , typename _Integer , typename _MonoidOperation >
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n,
    _MonoidOperation __monoid_op ) [inline]
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 113 of file ext/numeric.

**3.68.2.21 power()** [2/2]

```
template<typename _Tp , typename _Integer >
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n ) [inline]
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 123 of file ext/numeric.

**3.68.2.22 random\_sample()** [1/2]

```
template<typename _InputIterator , typename _RandomAccessIterator >
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last ) [inline]
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 388 of file ext/algorithm.

**3.68.2.23 random\_sample()** [2/2]

```
template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator >
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last,
    _RandomNumberGenerator & __rand ) [inline]
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 411 of file ext/algorithm.

**3.68.2.24 random\_sample\_n()** [1/2]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _Distance >
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n )
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 267 of file ext/algorithm.

**3.68.2.25 random\_sample\_n()** [2/2]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename _RandomNumberGenerator >
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n,
    _RandomNumberGenerator & __rand )
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 301 of file ext/algorithm.

### 3.68.2.26 uninitialized\_copy\_n()

```
template<typename _InputIter , typename _Size , typename _ForwardIter >
pair<_InputIter, _ForwardIter> __gnu_cxx::uninitialized_copy_n (
    _InputIter __first,
    _Size __count,
    _ForwardIter __result ) [inline]
```

Copies the range [first,last) into result.

#### Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	Length
<code>__result</code>	An output iterator.

#### Returns

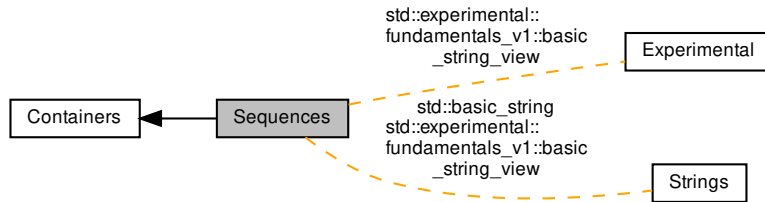
`__result + (__first + __count)`

Like `copy()`, but does not require an initialized output range.

Definition at line 122 of file `ext/memory`.

### 3.69 Sequences

Collaboration diagram for Sequences:



#### Classes

- struct `std::array<_Tp, _Nm>`
- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::deque<_Tp, _Alloc>`
- class `std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>`
- class `std::forward_list<_Tp, _Alloc>`
- class `std::list<_Tp, _Alloc>`
- class `std::priority_queue<_Tp, _Sequence, _Compare>`
- class `std::queue<_Tp, _Sequence>`
- class `std::stack<_Tp, _Sequence>`
- class `std::vector<_Tp, _Alloc>`
- class `std::vector<bool, _Alloc>`

#### 3.69.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

### 3.70 Set Operation

Collaboration diagram for Set Operation:



#### Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

#### 3.70.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.



### 3.70.2 Function Documentation

#### 3.70.2.1 `includes()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 >
bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2 ) [inline]
```

Determines whether all elements of a sequence exists in a range.

##### Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.

##### Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 2826 of file `stl_algo.h`.

#### 3.70.2.2 `includes()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare >
bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _Compare __comp ) [inline]
```

Determines whether all elements of a sequence exists in a range using comparison.

##### Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

**Returns**

True if each element in [`__first2`,`__last2`) is contained in order within [`__first1`,`__last1`) according to `comp`. False otherwise.

This operation expects both [`__first1`,`__last1`) and [`__first2`,`__last2`) to be sorted. Searches for the presence of each element in [`__first2`,`__last2`) within [`__first1`,`__last1`), using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [`__first2`,`__last2`) is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 2871 of file `stl_algo.h`.

**3.70.2.3 set\_difference()** [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
_OutputIterator std::set_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the difference of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5376 of file `stl_algo.h`.

### 3.70.2.4 `set_difference()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
_OutputIterator std::set_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Return the difference of two sorted ranges using comparison functor.

#### Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5427 of file `stl_algo.h`.

### 3.70.2.5 `set_intersection()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
_OutputIterator std::set_intersection (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the intersection of two sorted ranges.

## Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

## Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5254 of file `stl_algo.h`.

3.70.2.6 `set_intersection()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
_OutputIterator std::set_intersection (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Return the intersection of two sorted ranges using comparison functor.

## Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

## Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5303 of file `stl_algo.h`.

### 3.70.2.7 `set_symmetric_difference()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
_OutputIterator std::set_symmetric_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the symmetric difference of two sorted ranges.

#### Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

#### Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5506 of file `stl_algo.h`.

### 3.70.2.8 `set_symmetric_difference()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
_OutputIterator std::set_symmetric_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
```

```
_InputIterator2 __last2,  
_OutputIterator __result,  
_Compare __comp ) [inline]
```

Return the symmetric difference of two sorted ranges using comparison functor.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5557 of file `stl_algo.h`.

**3.70.2.9 set\_union()** [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
_OutputIterator std::set_union (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the union of two sorted ranges.

**Parameters**

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

**Returns**

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the

iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5133 of file `stl_algo.h`.

#### 3.70.2.10 `set_union()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
_OutputIterator std::set_union (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Return the union of two sorted ranges using a comparison functor.

##### Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

##### Returns

End of the output range.

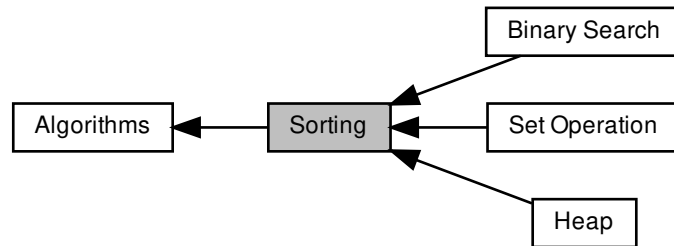
This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5183 of file `stl_algo.h`.



### 3.71 Sorting

Collaboration diagram for Sorting:



#### Modules

- [Binary Search](#)
- [Heap](#)
- [Set Operation](#)

#### Functions

- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::lexicographical\_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`

- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __↵`  
`last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __↵`  
`last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵`  
`Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵`  
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __↵`  
`last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __↵`  
`last, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b,`  
`_Compare __comp)`
- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator`  
`__first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator`  
`__first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccess↵`  
`Iterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccess↵`  
`Iterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccess↵`  
`Iterator __result_first, _RandomAccessIterator __result_last)`

- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

### 3.71.1 Detailed Description

### 3.71.2 Function Documentation

#### 3.71.2.1 `inplace_merge()` [1/2]

```
template<typename _BidirectionalIterator >
void std::inplace_merge (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last ) [inline]
```

Merges two sorted ranges in place.

#### Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

#### Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`__middle`,`__last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes  $(\text{__last} - \text{__first}) - 1$  comparisons. Otherwise an  $N \log N$  algorithm is used, where  $N$  is  $\text{distance}(\text{__first}, \text{__last})$ .

Definition at line 2574 of file `stl_algo.h`.

3.71.2.2 `inplace_merge()` [2/2]

```
template<typename _BidirectionalIterator, typename _Compare >
void std::inplace_merge (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    _Compare __comp ) [inline]
```

Merges two sorted ranges in place.

## Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A functor to use for comparisons.

## Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`__middle`,`__last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (`__last`-`__first`)-1 comparisons. Otherwise an NlogN algorithm is used, where N is `distance(__first, __last)`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2615 of file `stl_algo.h`.

3.71.2.3 `is_sorted()` [1/2]

```
template<typename _ForwardIterator >
bool std::is_sorted (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Determines whether the elements of a sequence are sorted.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

True if the elements are sorted, false otherwise.

Definition at line 3209 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**3.71.2.4 `is_sorted()`** [2/2]

```
template<typename _ForwardIterator, typename _Compare >
bool std::is_sorted (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Determines whether the elements of a sequence are sorted according to a comparison functor.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

True if the elements are sorted, false otherwise.

Definition at line 3223 of file `stl_algo.h`.

References `std::is_sorted_until()`.

**3.71.2.5 `is_sorted_until()`** [1/2]

```
template<typename _ForwardIterator >
_FowardIterator std::is_sorted_until (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Determines the end of a sorted sequence.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

An iterator pointing to the last iterator *i* in [*\_\_first*, *\_\_last*) for which the range [*\_\_first*, *i*) is sorted.

Definition at line 3252 of file `stl_algo.h`.

**3.71.2.6 `is_sorted_until()`** [2/2]

```
template<typename _ForwardIterator , typename _Compare >
_FForwardIterator std::is_sorted_until (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Determines the end of a sorted sequence using comparison functor.

**Parameters**

<i>__first</i>	An iterator.
<i>__last</i>	Another iterator.
<i>__comp</i>	A comparison functor.

**Returns**

An iterator pointing to the last iterator *i* in [*\_\_first*, *\_\_last*) for which the range [*\_\_first*, *i*) is sorted.

Definition at line 3276 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

**3.71.2.7 `lexicographical_compare()`** [1/2]

```
template<typename _II1 , typename _II2 >
bool std::lexicographical_compare (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2 ) [inline]
```

Performs **dictionary** comparison on ranges.

**Parameters**

<i>__first1</i>	An input iterator.
<i>__last1</i>	An input iterator.
<i>__first2</i>	An input iterator.
<i>__last2</i>	An input iterator.

**Returns**

A boolean true or false.

*Returns true if the sequence of elements defined by the range `[first1,last1)` is lexicographically less than the sequence of elements defined by the range `[first2,last2)`. Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1221 of file `stl_algobase.h`.

**3.71.2.8 lexicographical\_compare()** [2/2]

```
template<typename _II1 , typename _II2 , typename _Compare >
bool std::lexicographical_compare (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2,
    _Compare __comp ) [inline]
```

Performs **dictionary** comparison on ranges.

**Parameters**

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A <a href="#">comparison functor</a> .

**Returns**

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1257 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

**3.71.2.9 max()** [1/2]

```
template<typename _Tp >
_GLIBCXX14_CONSTEXPR const _Tp & std::max (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

This does what you think it does.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

## Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 219 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_nth_element()`, `std::deque<_StateSeqT>::_M_initialize_map()`, `std::deque<_StateSeqT>::_M_reallocate_map()`, `std::discard_block_engine<_RandomNumberEngine, __p, __r>::max()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::max()`, and `__gnu_parallel::multiseq_selection()`.

3.71.2.10 `max()` [2/2]

```
template<typename _Tp, typename _Compare>
_GLIBCXX14_CONSTEXPR const _Tp & std::max (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp ) [inline]
```

This does what you think it does.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A <a href="#">comparison functor</a> .

## Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 265 of file `stl_algobase.h`.



**3.71.2.11** `max_element()` [1/2]

```
template<typename _ForwardIterator >
_GLIBCXX14_CONSTEXPR _ForwardIterator std::max_element (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Return the maximum element in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

Iterator referencing the first instance of the largest value.

Definition at line 5674 of file `stl_algo.h`.

**3.71.2.12** `max_element()` [2/2]

```
template<typename _ForwardIterator , typename _Compare >
_GLIBCXX14_CONSTEXPR _ForwardIterator std::max_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Return the maximum element in a range using comparison functor.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

**Returns**

Iterator referencing the first instance of the largest value according to `__comp`.

Definition at line 5699 of file `stl_algo.h`.

3.71.2.13 `merge()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
_OutputIterator std::merge (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Merges two sorted ranges.

## Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.

## Returns

An iterator pointing to the first element *not less than* *val*.

Merges the ranges `[__first1,__last1)` and `[__first2,__last2)` into the sorted range `[__result, __result + (__last1-__first1) + (__last2-__first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 4916 of file `stl_algo.h`.

3.71.2.14 `merge()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
_OutputIterator std::merge (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Merges two sorted ranges.

## Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.
<code>__comp</code>	A functor to use for comparisons.

**Returns**

An iterator pointing to the first element "not less than" *val*.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 4966 of file `stl_algo.h`.

**3.71.2.15 min()** [1/2]

```
template<typename _Tp >
_GLIBCXX14_CONSTEXPR const _Tp & std::min (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

This does what you think it does.

**Parameters**

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

**Returns**

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 195 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort_qs_divide()`, `std::__sample()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `std::basic_string<char>::compare()`, `std::generate_canonical()`, `std::discard_block_engine<_RandomNumberEngine, __p, __r>::min()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::min()`, `std::basic_streambuf<_Elem, _Tr>::xsgetn()`, and `std::basic_streambuf<_Elem, _Tr>::xsputn()`.

**3.71.2.16 min()** [2/2]

```
template<typename _Tp , typename _Compare >
_GLIBCXX14_CONSTEXPR const _Tp & std::min (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp ) [inline]
```

This does what you think it does.

**Parameters**

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A <a href="#">comparison functor</a> .

**Returns**

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 243 of file `stl_algobase.h`.

**3.71.2.17 min\_element()** [1/2]

```
template<typename _ForwardIterator >
_GLIBCXX14_CONSTEXPR _ForwardIterator std::min_element (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Return the minimum element in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

Iterator referencing the first instance of the smallest value.

Definition at line 5610 of file `stl_algo.h`.

**3.71.2.18 min\_element()** [2/2]

```
template<typename _ForwardIterator, typename _Compare>
_GLIBCXX14_CONSTEXPR _ForwardIterator std::min_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Return the minimum element in a range using comparison functor.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

**Returns**

Iterator referencing the first instance of the smallest value according to `__comp`.

Definition at line 5635 of file `stl_algo.h`.

**3.71.2.19 minmax()** [1/2]

```
template<typename _Tp>
_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

Determines min and max at once as an ordered pair.

**Parameters**

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

**Returns**

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 3302 of file `stl_algo.h`.

## 3.71.2.20 minmax() [2/2]

```
template<typename _Tp , typename _Compare >
_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp ) [inline]
```

Determines min and max at once as an ordered pair.

## Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A <a href="#">comparison functor</a> .

## Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 3323 of file `stl_algo.h`.

## 3.71.2.21 minmax\_element() [1/2]

```
template<typename _ForwardIterator >
_GLIBCXX14_CONSTEXPR pair<_ForwardIterator, _ForwardIterator> std::minmax_element (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

## Returns

make\_pair(`m`, `M`), where `m` is the first iterator `i` in `[__first, __last)` such that no other element in the range is smaller, and where `M` is the last iterator `i` in `[__first, __last)` such that no other element in the range is larger.

Definition at line 3403 of file `stl_algo.h`.

**3.71.2.22 minmax\_element()** [2/2]

```
template<typename _ForwardIterator , typename _Compare >
_GLIBCXX14_CONSTEXPR pair<_ForwardIterator, _ForwardIterator> std::minmax_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

**Returns**

make\_pair(m, M), where m is the first iterator i in [`__first`, `__last`) such that no other element in the range is smaller, and where M is the last iterator i in [`__first`, `__last`) such that no other element in the range is larger.

Definition at line 3431 of file `stl_algo.h`.

**3.71.2.23 next\_permutation()** [1/2]

```
template<typename _BidirectionalIterator >
bool std::next_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Permute range into the next *dictionary* ordering.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 2954 of file `stl_algo.h`.

3.71.2.24 `next_permutation()` [2/2]

```
template<typename _BidirectionalIterator, typename _Compare>
bool std::next_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Compare __comp ) [inline]
```

Permute range into the next *dictionary* ordering using comparison functor.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

## Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range `[__first,__last)` as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 2986 of file `stl_algo.h`.

3.71.2.25 `nth_element()` [1/2]

```
template<typename _RandomAccessIterator>
void std::nth_element (
    _RandomAccessIterator __first,
    _RandomAccessIterator __nth,
    _RandomAccessIterator __last ) [inline]
```

Sort a sequence just enough to find a particular position.

## Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.



Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `*j < *i` is false.

Definition at line 4748 of file `stl_algo.h`.

#### 3.71.2.26 `nth_element()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
void std::nth_element (
    _RandomAccessIterator __first,
    _RandomAccessIterator __nth,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort a sequence just enough to find a particular position using a predicate for comparison.

##### Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

##### Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `__comp(*j,*i)` is false.

Definition at line 4787 of file `stl_algo.h`.

#### 3.71.2.27 `partial_sort()` [1/2]

```
template<typename _RandomAccessIterator >
void std::partial_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last ) [inline]
```

Sort the smallest elements of a sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[first,last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*j < *i` and `*k < *i` are both false.

Definition at line 4674 of file `stl_algo.h`.

3.71.2.28 `partial_sort()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare >
void std::partial_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort the smallest elements of a sequence using a predicate for comparison.

## Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

## Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[__first,__last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*__comp(j,*i)` and `*__comp(*k,*i)` are both false.

Definition at line 4712 of file `stl_algo.h`.

3.71.2.29 `partial_sort_copy()` [1/2]

```
template<typename _InputIterator , typename _RandomAccessIterator >
_RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __result_first,
    _RandomAccessIterator __result_last ) [inline]
```

Copy the smallest elements of a sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.

## Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest  $N$  values from the range  $[\text{__first}, \text{__last})$  to the range beginning at `__result_first`, where the number of elements to be copied,  $N$ , is the smaller of  $(\text{__last} - \text{__first})$  and  $(\text{__result\_last} - \text{__result\_first})$ . After the sort if  $i$  and  $j$  are iterators in the range  $[\text{__result\_first}, \text{__result\_first} + N)$  such that  $i$  precedes  $j$  then  $*j < *i$  is false. The value returned is `__result_first + N`.

Definition at line 1737 of file `stl_algo.h`.

3.71.2.30 `partial_sort_copy()` [2/2]

```
template<typename _InputIterator , typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __result_first,
    _RandomAccessIterator __result_last,
    _Compare __comp ) [inline]
```

Copy the smallest elements of a sequence using a predicate for comparison.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	Another input iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.
<code>__comp</code>	A comparison functor.

**Returns**

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest  $N$  values from the range  $[\_\text{first}, \_\text{last})$  to the range beginning at `result\_first`, where the number of elements to be copied,  $N$ , is the smaller of  $(\_\text{last} - \_\text{first})$  and  $(\_\text{result\_last} - \_\text{result\_first})$ . After the sort if  $i$  and  $j$  are iterators in the range  $[\_\text{result\_first}, \_\text{result\_first} + N)$  such that  $i$  precedes  $j$  then `__comp(*j, *i)` is false. The value returned is `__result\_first + N`.

Definition at line 1787 of file `stl_algo.h`.

**3.71.2.31 prev\_permutation()** [1/2]

```
template<typename _BidirectionalIterator >
bool std::prev_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Permute range into the previous *dictionary* ordering.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

**Returns**

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3054 of file `stl_algo.h`.

**3.71.2.32 prev\_permutation()** [2/2]

```
template<typename _BidirectionalIterator , typename _Compare >
bool std::prev_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Compare __comp ) [inline]
```

Permute range into the previous *dictionary* ordering using comparison functor.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

## Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range [`first`,`last`) as a set of *dictionary* sorted sequences ordered by `comp`. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3086 of file stl\_algo.h.

### 3.71.2.33 `sort()` [1/2]

```
template<typename _RandomAccessIterator >
void std::sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

## Sort the elements of a sequence.

## Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

## Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator *i* in the range `[__first,__last-1)`, `*i+1 < *i` is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 4824 of file stl\_algo.h.

**3.71.2.34** `sort()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

**Returns**

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that `__comp(*(i+1),*i)` is false for every iterator `i` in the range `[__first,__last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 4854 of file `stl_algo.h`.

**3.71.2.35** `stable_sort()` [1/2]

```
template<typename _RandomAccessIterator>
void std::stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Sort the elements of a sequence, preserving the relative order of equivalent elements.

**Parameters**

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

**Returns**

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator `i` in the range `[__first,__last-1)`, `*(i+1) < *i` is false.

The relative ordering of equivalent elements is preserved, so any two elements  $x$  and  $y$  in the range  $[\_\_first, \_\_last)$  such that  $x < y$  is false and  $y < x$  is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5029 of file `stl_algo.h`.

### 3.71.2.36 `stable_sort()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare >
void std::stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

#### Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

#### Returns

Nothing.

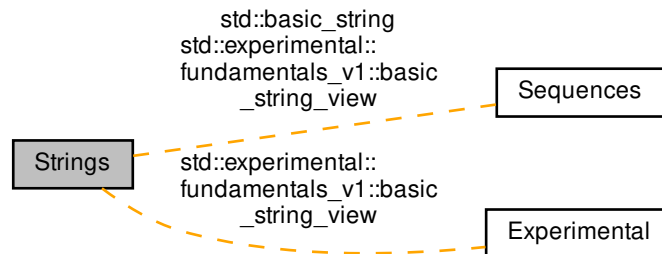
Sorts the elements in the range  $[\_\_first, \_\_last)$  in ascending order, such that for each iterator  $i$  in the range  $[\_\_first, \_\_last-1)$ , `__comp(*(i+1),*i)` is false.

The relative ordering of equivalent elements is preserved, so any two elements  $x$  and  $y$  in the range  $[\_\_first, \_\_last)$  such that `__comp(x, y)` is false and `__comp(y, x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5063 of file `stl_algo.h`.

## 3.72 Strings

Collaboration diagram for Strings:



### Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`
- struct `std::char_traits< _CharT >`
- class `std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >`

### Typedefs

- typedef `basic_string< char >` `std::string`
- typedef `basic_string< char16_t >` `std::u16string`
- typedef `basic_string< char32_t >` `std::u32string`
- typedef `basic_string< wchar_t >` `std::wstring`

### 3.72.1 Detailed Description

### 3.72.2 Typedef Documentation

#### 3.72.2.1 string

```
typedef basic_string<char> std::string
```

A string of `char`.

Definition at line 71 of file `stringfwd.h`.



#### 3.72.2.2 u16string

```
typedef basic\_string<char16_t> std::u16string
```

A string of `char16_t`.

Definition at line 84 of file `stringfwd.h`.

#### 3.72.2.3 u32string

```
typedef basic\_string<char32_t> std::u32string
```

A string of `char32_t`.

Definition at line 87 of file `stringfwd.h`.

#### 3.72.2.4 wstring

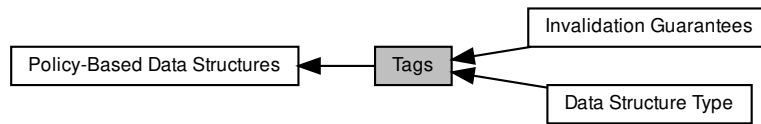
```
typedef basic\_string<wchar_t> std::wstring
```

A string of `wchar_t`.

Definition at line 78 of file `stringfwd.h`.

### 3.73 Tags

Collaboration diagram for Tags:



#### Modules

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

#### Classes

- [struct `\_\_gnu\_pbds::trivial\_iterator\_tag`](#)

#### Typedefs

- [typedef void `\_\_gnu\_pbds::trivial\_iterator\_difference\_type`](#)

#### 3.73.1 Detailed Description

#### 3.73.2 Typedef Documentation

##### 3.73.2.1 `trivial_iterator_difference_type`

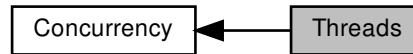
```
typedef void \_\_gnu\_pbds::trivial\_iterator\_difference\_type
```

Prohibit moving trivial iterators.

Definition at line 79 of file `tag_and_trait.hpp`.

### 3.74 Threads

Collaboration diagram for Threads:



#### Namespaces

- [std::this\\_thread](#)

#### Classes

- struct [std::hash< thread::id >](#)
- class [std::thread](#)

#### Functions

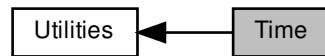
- bool **std::operator!=** ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- bool **std::operator<** ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- template<class \_CharT, class \_Traits >  
[basic\\_ostream](#)< \_CharT, \_Traits > & **std::operator<<** ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_out, [thread::id](#) \_\_id)
- bool **std::operator<=** ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- bool **std::operator==** ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- bool **std::operator>** ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- bool **std::operator>=** ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- void **std::swap** ([thread](#) &\_\_x, [thread](#) &\_\_y) noexcept

#### 3.74.1 Detailed Description

Classes for thread support.

### 3.75 Time

Collaboration diagram for Time:



#### Namespaces

- [std::chrono](#)

#### Macros

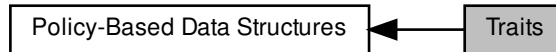
- `#define __cpp_lib_chrono_udls`

#### 3.75.1 Detailed Description

Classes and functions for time.

### 3.76 Traits

Collaboration diagram for Traits:



#### Classes

- struct `__gnu_pbds::container_traits< Cntnr >`
- struct `__gnu_pbds::container_traits_base< _Tag >`
- struct `__gnu_pbds::container_traits_base< binary_heap_tag >`
- struct `__gnu_pbds::container_traits_base< binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< cc_hash_tag >`
- struct `__gnu_pbds::container_traits_base< gp_hash_tag >`
- struct `__gnu_pbds::container_traits_base< list_update_tag >`
- struct `__gnu_pbds::container_traits_base< ov_tree_tag >`
- struct `__gnu_pbds::container_traits_base< pairing_heap_tag >`
- struct `__gnu_pbds::container_traits_base< pat_trie_tag >`
- struct `__gnu_pbds::container_traits_base< rb_tree_tag >`
- struct `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< splay_tree_tag >`
- struct `__gnu_pbds::container_traits_base< thin_heap_tag >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, Mapped >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, null_type >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th >`
- struct `__gnu_pbds::detail::stored_data< _Tv, null_type >`
- struct `__gnu_pbds::detail::stored_hash< _Th >`
- struct `__gnu_pbds::detail::stored_value< _Tv >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`

- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`
- struct `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`
- struct `__gnu_pbds::null_type`

#### Variables

- static `null_type` `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >::s_null_type`
- static `null_type` `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >::s_null_type`

#### 3.76.1 Detailed Description

### 3.77 Type-safe container of any type

Collaboration diagram for Type-safe container of any type:



#### Classes

- class `std::experimental::fundamentals_v1::any`
- class `std::experimental::fundamentals_v1::bad_any_cast`

#### Macros

- `#define __cpp_lib_experimental_any`

#### Functions

- void `std::experimental::fundamentals_v1::__throw_bad_any_cast()`
- static void `std::experimental::fundamentals_v1::any::Manager_internal<_Tp>::_S_manage(_Op __op, const any *__anyp, _Arg *__arg)`
- static void `std::experimental::fundamentals_v1::any::Manager_external<_Tp>::_S_manage(_Op __op, const any *__anyp, _Arg *__arg)`
- template<typename \_ValueType>  
\_ValueType `std::experimental::fundamentals_v1::any_cast(const any &__any)`
- void `std::experimental::fundamentals_v1::swap(any &__x, any &__y) noexcept`
- template<typename \_ValueType>  
\_ValueType `std::experimental::fundamentals_v1::any_cast(any &__any)`
- template<typename \_ValueType, typename enable\_if<is\_move\_constructible<\_ValueType>::value||is\_lvalue\_reference<\_ValueType>::value, bool>::type = true>  
\_ValueType `std::experimental::fundamentals_v1::any_cast(any &&__any)`
- template<typename \_ValueType>  
const \_ValueType \* `std::experimental::fundamentals_v1::any_cast(const any *__any) noexcept`
- template<typename \_ValueType>  
\_ValueType \* `std::experimental::fundamentals_v1::any_cast(any *__any) noexcept`

#### 3.77.1 Detailed Description

A type-safe container for single values of value types, as described in n3804 "Any Library Proposal (Revision 3)".

## 3.77.2 Function Documentation

3.77.2.1 `any_cast()` [1/5]

```
template<typename _ValueType >
_ValueType std::experimental::fundamentals_v1::any_cast (
    const any & __any ) [inline]
```

Access the contained object.

## Template Parameters

<code>_ValueType</code>	A const-reference or CopyConstructible type.
-------------------------	--

## Parameters

<code>__any</code>	The object to access.
--------------------	-----------------------

## Returns

The contained object.

## Exceptions

<code>bad_any_cast</code>	If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code>
---------------------------	--

Definition at line 352 of file `any`.

3.77.2.2 `any_cast()` [2/5]

```
template<typename _ValueType >
_ValueType std::experimental::fundamentals_v1::any_cast (
    any & __any ) [inline]
```

Access the contained object.

## Template Parameters

<code>_ValueType</code>	A reference or CopyConstructible type.
-------------------------	--



**Parameters**

<code>__any</code>	The object to access.
--------------------	-----------------------

**Returns**

The contained object.

**Exceptions**

<i>bad_any_cast</i>	If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code>
---------------------	--

Definition at line 375 of file any.

**3.77.2.3 any\_cast()** [3/5]

```
template<typename _ValueType , typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
_ValueType std::experimental::fundamentals_v1::any_cast (
    any && __any ) [inline]
```

Access the contained object.

**Template Parameters**

<code>_ValueType</code>	A reference or CopyConstructible type.
-------------------------	--

**Parameters**

<code>__any</code>	The object to access.
--------------------	-----------------------

**Returns**

The contained object.

**Exceptions**

<i>bad_any_cast</i>	If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code>
---------------------	--

Definition at line 389 of file any.

3.77.2.4 `any_cast()` [4/5]

```
template<typename _ValueType >
const _ValueType* std::experimental::fundamentals_v1::any_cast (
    const any * __any ) [inline], [noexcept]
```

Access the contained object.

## Template Parameters

<code>_ValueType</code>	The type of the contained object.
-------------------------	-----------------------------------

## Parameters

<code>__any</code>	A pointer to the object to access.
--------------------	------------------------------------

## Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

Definition at line 465 of file any.

3.77.2.5 `any_cast()` [5/5]

```
template<typename _ValueType >
_ValueType* std::experimental::fundamentals_v1::any_cast (
    any * __any ) [inline], [noexcept]
```

Access the contained object.

## Template Parameters

<code>_ValueType</code>	The type of the contained object.
-------------------------	-----------------------------------

## Parameters

<code>__any</code>	A pointer to the object to access.
--------------------	------------------------------------

## Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

Definition at line 473 of file any.

### 3.77.2.6 swap()

```
void std::experimental::fundamentals_v1::swap (
    any & __x,
    any & __y ) [inline], [noexcept]
```

Exchange the states of two `any` objects.

Definition at line 339 of file `any`.

## 3.78 Uniform Distributions

Collaboration diagram for Uniform Distributions:



### Classes

- class `std::uniform_real_distribution<_RealType>`

### Functions

- `template<typename _IntType>`  
`bool std::operator!= (const std::uniform_int_distribution<_IntType> &__d1, const std::uniform_int_distribution<_IntType> &__d2)`
- `template<typename _IntType>`  
`bool std::operator!= (const std::uniform_real_distribution<_IntType> &__d1, const std::uniform_real_distribution<_IntType> &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_int_distribution<_IntType> &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_real_distribution<_RealType> &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_int_distribution<_IntType> &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_real_distribution<_RealType> &)`

### 3.78.1 Detailed Description

### 3.78.2 Function Documentation

**3.78.2.1 operator!=()** [1/2]

```
template<typename _IntType >
bool std::operator!= (
    const std::uniform_int_distribution< _IntType > & __d1,
    const std::uniform_int_distribution< _IntType > & __d2 ) [inline]
```

Return true if two uniform integer distributions have different parameters.

Definition at line 1660 of file random.h.

**3.78.2.2 operator!=()** [2/2]

```
template<typename _IntType >
bool std::operator!= (
    const std::uniform_real_distribution< _IntType > & __d1,
    const std::uniform_real_distribution< _IntType > & __d2 ) [inline]
```

Return true if two uniform real distributions have different parameters.

Definition at line 1874 of file random.h.

**3.78.2.3 operator<<()** [1/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::uniform_int_distribution< _IntType > & __x )
```

Inserts a uniform\_int\_distribution random number distribution \_\_x into the output stream os.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A uniform_int_distribution random number distribution.

**Returns**

The output stream with the state of \_\_x inserted or in an error state.

Definition at line 874 of file bits/random.tcc.

References `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::scientific()`, and `std::basic_ostream< _CharT, _Traits >::widen()`.

3.78.2.4 `operator<<()` [2/2]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::uniform_real_distribution< _RealType > & __x )
```

Inserts a `uniform_real_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 933 of file `bits/random.tcc`.

References `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::left()`, `std::ios_base::precision()`, `std::scientific()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

3.78.2.5 `operator>>()` [1/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::uniform_int_distribution< _IntType > & __x )
```

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

Definition at line 895 of file `bits/random.tcc`.



References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution< _IntType >::param()`, and `std::skipws()`.

### 3.78.2.6 operator>>() [2/2]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::uniform_real_distribution< _RealType > & __x )
```

Extracts a uniform\_real\_distribution random number distribution \_\_x from the input stream \_\_is.

#### Parameters

 __is	An input stream.
 __x	A uniform_real_distribution random number generator engine.

#### Returns

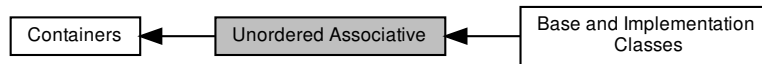
The input stream with \_\_x extracted or in an error state.

Definition at line 957 of file bits/random.tcc.

References std::ios\_base::flags(), std::uniform\_real\_distribution< \_RealType >::param(), and std::skipws().

### 3.79 Unordered Associative

Collaboration diagram for Unordered Associative:



#### Modules

- [Base and Implementation Classes](#)

#### Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

#### 3.79.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

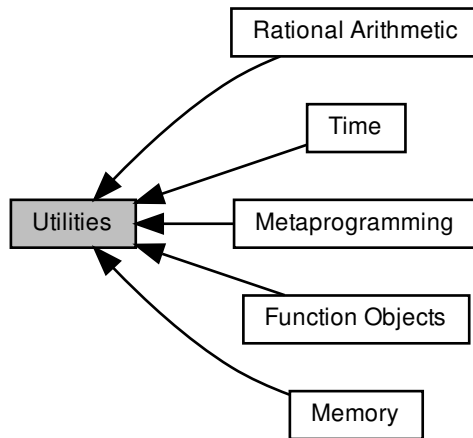
Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).



### 3.80 Utilities

Collaboration diagram for Utilities:



#### Modules

- [Function Objects](#)
- [Memory](#)
- [Metaprogramming](#)
- [Rational Arithmetic](#)
- [Time](#)

#### Classes

- `struct std::_Tuple_impl< _Idx, _Elements >`
- `struct std::_Tuple_impl< _Idx, _Head, _Tail... >`
- `class std::bitset< _Nb >`
- `struct std::pair< _T1, _T2 >`
- `struct std::piecewise_construct_t`
- `class std::tuple< _Elements >`
- `class std::tuple< _T1, _T2 >`
- `struct std::tuple_element< 0, tuple< _Head, _Tail... > >`
- `struct std::tuple_element< __i, tuple< _Head, _Tail... > >`
- `struct std::tuple_element< __i, tuple<> >`
- `struct std::tuple_size< tuple< _Elements... > >`
- `struct std::type_index`
- `struct std::uses_allocator< tuple< _Types... >, _Alloc >`

## Macros

- `#define __cpp_lib_tuples_by_type`

## Typedefs

- `template<typename _Tp >`  
`using std::__empty_not_final = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp > >::type`

## Functions

- `template<typename... _Args1, typename... _Args2>`  
`std::pair< _T1, _T2 >::pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)`
- `template<typename _Tp >`  
`constexpr _Tp * std::__addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`  
`_Tp std::__exchange (_Tp &__obj, _Up &&__new_val)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & std::__get_helper (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & std::__get_helper (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr _Head & std::__get_helper2 (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr const _Head & std::__get_helper2 (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`  
`constexpr _Up && std::__invfwd (typename remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result< _Callable, _Args... >::type std::__invoke (_Callable &&__fn, _Args &&... __args) noexcept( __is_nothrow_invocable< _Callable, _Args... >::value)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args) ←`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Tp >`  
`_GLIBCXX17_CONSTEXPR _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`  
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &&__t) noexcept`

- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > std::forward_as_tuple ( _Elements &&... __args) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp & std::get (tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp && std::get (tuple< _Types... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp & std::get (const tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp && std::get (const tuple< _Types... > &&__t) noexcept`
- `template<typename _T1, typename _T2 >`  
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > > std::make_pair ( _T1 &&__x, _T2 &&__y)`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple ( _Elements &&... __args)`
- `template<typename _Tp >`  
`constexpr std::remove_reference< _Tp >::type && std::move ( _Tp &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type std::move_if_noexcept ( _Tp &__x) noexcept`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator!= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator< (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator<= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator== (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator> (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`

- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator>= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _T1, typename _T2 >`  
`enable\_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`enable\_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp >`  
`enable\_if< __and< __not< __is_tuple_like< _Tp > >, is\_move\_constructible< _Tp >, is\_move\_assignable< _Tp > >::value >::type std::swap (_Tp &__a, _Tp &__b) noexcept(__and< is\_nothrow\_move\_constructible< _Tp >, is\_nothrow\_move\_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm>`  
`enable\_if< __is_swappable< _Tp >::value >::type std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept`
- `template<typename... _Tpls, typename = typename enable\_if< __and< __is_tuple_like< _Tpls >... >::value >::type >`  
`constexpr auto std::tuple\_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type`

## Variables

- `_GLIBCXX17_INLINE constexpr _Swallow_assign std::ignore`
- `_GLIBCXX17_INLINE constexpr piecewise\_construct\_t std::piecewise\_construct`

### 3.80.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

### 3.80.2 Function Documentation

#### 3.80.2.1 `__addressof()`

```
template<typename _Tp >
constexpr _Tp* std::\_\_addressof (
    _Tp & __r ) [inline], [noexcept]
```

Same as C++11 `std::addressof`.

Definition at line 47 of file `move.h`.

Referenced by `std::_Destroy()`, `std::__addressof()`, `std::list< __inp, __rebind_inp >::merge()`, `std::reverse_iterator< __inp, __rebind_inp >::operator->()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::list< __inp, __rebind_inp >::operator=()`, `std::list< __inp, __rebind_inp >::remove()`, `std::rethrow_if_nested()`, and `std::list< __inp, __rebind_inp >::splice()`.

### 3.80.2.2 `__invoke()`

```
template<typename _Callable , typename... _Args>
constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (
    _Callable && __fn,
    _Args &&... __args ) [noexcept]
```

Invoke a callable object.

Definition at line 89 of file `invoke.h`.

### 3.80.2.3 `addressof()`

```
template<typename _Tp >
_GLIBCXX17_CONSTEXPR _Tp* std::addressof (
    _Tp & __r ) [inline], [noexcept]
```

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

#### Parameters

<code>↔</code>	Reference to an object or function.
<code>__↔</code>	
<code>↔</code>	
<code>__↔</code>	
<code>r</code>	

#### Returns

The actual address.

Definition at line 138 of file `move.h`.

References `std::__addressof()`.

Referenced by `std::pointer_traits<_Tp*>::pointer_to()`.

### 3.80.2.4 `forward()` [1/2]

```
template<typename _Tp >
constexpr _Tp&& std::forward (
    typename std::remove_reference<_Tp>::type & __t ) [noexcept]
```

Forward an lvalue.

#### Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 74 of file `move.h`.

### 3.80.2.5 forward() [2/2]

```
template<typename _Tp >
constexpr _Tp&& std::forward (
    typename std::remove_reference< _Tp >::type && __t ) [noexcept]
```

Forward an rvalue.

#### Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 85 of file move.h.

### 3.80.2.6 get() [1/8]

```
template<std::size_t __i, typename... _Elements>
constexpr __tuple_element_t<__i, tuple<_Elements...> >& std::get (
    tuple< _Elements... > & __t ) [noexcept]
```

Return a reference to the ith element of a tuple.

Definition at line 1314 of file tuple.

### 3.80.2.7 get() [2/8]

```
template<std::size_t __i, typename... _Elements>
constexpr const __tuple_element_t<__i, tuple<_Elements...> >& std::get (
    const tuple< _Elements... > & __t ) [noexcept]
```

Return a const reference to the ith element of a const tuple.

Definition at line 1320 of file tuple.

### 3.80.2.8 get() [3/8]

```
template<std::size_t __i, typename... _Elements>
constexpr __tuple_element_t<__i, tuple<_Elements...> >&& std::get (
    tuple< _Elements... > && __t ) [noexcept]
```

Return an rvalue reference to the ith element of a tuple rvalue.

Definition at line 1326 of file tuple.

**3.80.2.9** `get()` [4/8]

```
template<std::size_t __i, typename... _Elements>
constexpr const __tuple_element_t<__i, tuple<_Elements...> >&& std::get (
    const tuple<_Elements...> && __t ) [noexcept]
```

Return a const rvalue reference to the *i*th element of a const tuple rvalue.

Definition at line 1335 of file tuple.

**3.80.2.10** `get()` [5/8]

```
template<typename _Tp, typename... _Types>
constexpr _Tp& std::get (
    tuple<_Types...> & __t ) [noexcept]
```

Return a reference to the unique element of type *\_Tp* of a tuple.

Definition at line 1358 of file tuple.

**3.80.2.11** `get()` [6/8]

```
template<typename _Tp, typename... _Types>
constexpr _Tp&& std::get (
    tuple<_Types...> && __t ) [noexcept]
```

Return a reference to the unique element of type *\_Tp* of a tuple rvalue.

Definition at line 1364 of file tuple.

**3.80.2.12** `get()` [7/8]

```
template<typename _Tp, typename... _Types>
constexpr const _Tp& std::get (
    const tuple<_Types...> & __t ) [noexcept]
```

Return a const reference to the unique element of type *\_Tp* of a tuple.

Definition at line 1370 of file tuple.

**3.80.2.13** `get()` [8/8]

```
template<typename _Tp , typename... _Types>
constexpr const _Tp&& std::get (
    const tuple< _Types... > && __t ) [noexcept]
```

Return a const reference to the unique element of type `_Tp` of a const tuple rvalue.

Definition at line 1377 of file `tuple`.

**3.80.2.14** `make_pair()`

```
template<typename _T1 , typename _T2 >
constexpr pair<typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type>
std::make_pair (
    _T1 && __x,
    _T2 && __y )
```

A convenience wrapper for creating a pair from two objects.

**Parameters**

<code>__x</code>	The first object.
<code>__y</code>	The second object.

**Returns**

A newly-constructed `pair<>` object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 524 of file `stl_pair.h`.

Referenced by `std::__gen_two_uniform_ints()`, `__gnu_debug::__get_distance()`, `__gnu_parallel::__parallel_merge<>_advance()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_debug::__valid_range_aux()`, `__gnu_parallel::__find<>_if_selector::__M_sequential_algorithm()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential<>_algorithm()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__parallel_sort_mwms_pu()`, and `__gnu_pbds::detail::__pat_trie_base::__Node_citer<Node, Leaf, Head, Inode, Clterator, Iterator, _Alloc>::__valid_prefix()`.

**3.80.2.15** `move()`

```
template<typename _Tp >
constexpr std::remove_reference<_Tp>::type&& std::move (
    _Tp && __t ) [noexcept]
```

Convert a value to an rvalue.



**Parameters**

↔	A thing of arbitrary type.
↔	
↔	
↔	
<i>t</i>	

**Returns**

The parameter cast to an rvalue-reference to allow moving it.

Definition at line 99 of file move.h.

**3.80.2.16 move\_if\_noexcept()**

```
template<typename _Tp >
constexpr conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp&, _Tp&&>::type std::move_↔
if_noexcept (
    _Tp & __x ) [noexcept]
```

Conditionally convert a value to an rvalue.

**Parameters**

↔	A thing of arbitrary type.
<i>x</i>	

**Returns**

The parameter, possibly cast to an rvalue-reference.

Same as std::move unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Definition at line 119 of file move.h.

**3.80.2.17 operator!=(())**

```
template<typename _T1 , typename _T2 >
constexpr bool std::operator!= (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [inline]
```

Uses operator== to find the result.

Definition at line 461 of file stl\_pair.h.

### 3.80.2.18 operator<()

```
template<typename _T1 , typename _T2 >
constexpr bool std::operator< (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [inline]
```

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

Definition at line 454 of file stl\_pair.h.

References `std::pair<_T1, _T2>::first`, and `std::pair<_T1, _T2>::second`.

### 3.80.2.19 operator<=()

```
template<typename _T1 , typename _T2 >
constexpr bool std::operator<= (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [inline]
```

Uses `operator<` to find the result.

Definition at line 473 of file stl\_pair.h.

### 3.80.2.20 operator==()

```
template<typename _T1 , typename _T2 >
constexpr bool std::operator== (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [inline]
```

Two pairs of the same type are equal iff their members are equal.

Definition at line 448 of file stl\_pair.h.

References `std::pair<_T1, _T2>::first`, and `std::pair<_T1, _T2>::second`.

### 3.80.2.21 operator>()

```
template<typename _T1 , typename _T2 >
constexpr bool std::operator> (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [inline]
```

Uses `operator<` to find the result.

Definition at line 467 of file stl\_pair.h.

**3.80.2.22 operator>=()**

```
template<typename _T1 , typename _T2 >
constexpr bool std::operator>= (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [inline]
```

Uses `operator<` to find the result.

Definition at line 479 of file `stl_pair.h`.

**3.80.2.23 swap()** [1/4]

```
template<typename _T1 , typename _T2 >
enable_if<__and<__is_swappable<_T1>, __is_swappable<_T2> >::value>::type std::swap (
    pair< _T1, _T2 > & __x,
    pair< _T1, _T2 > & __y ) [inline], [delete], [noexcept]
```

See `std::pair::swap()`.

Definition at line 495 of file `stl_pair.h`.

**3.80.2.24 swap()** [2/4]

```
template<typename... _Elements>
enable_if<__and<__is_swappable<_Elements>...>::value >::type std::swap (
    tuple< _Elements... > & __x,
    tuple< _Elements... > & __y ) [inline], [delete], [noexcept]
```

`swap`

Definition at line 1618 of file `tuple`.

**3.80.2.25 swap()** [3/4]

```
template<typename _Tp >
enable_if< __and< __not< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable<
_Tp > >::value >::type std::swap (
    _Tp & __a,
    _Tp & __b ) const [inline], [noexcept]
```

Swaps two values.

## Parameters

$\leftrightarrow$ _a	A thing of arbitrary type.
$\leftrightarrow$ _b	Another thing of arbitrary type.

## Returns

Nothing.

Definition at line 182 of file move.h.

## 3.80.2.26 swap() [4/4]

```
template<typename _Tp , size_t _Nm>
enable_if< __is_swappable< _Tp >::value >::type std::swap (
    _Tp(&) __a[_Nm],
    _Tp(&) __b[_Nm] ) [inline], [noexcept]
```

Swap the contents of two arrays.

Definition at line 205 of file move.h.

## 3.80.2.27 tie()

```
template<typename... _Elements>
constexpr tuple<_Elements&...> std::tie (
    _Elements &... __args ) [noexcept]
```

tie

Definition at line 1605 of file tuple.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 3.80.2.28 tuple\_cat()

```
template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>
constexpr auto std::tuple_cat (
    _Tpls &&... __tpls ) -> typename __tuple_cat_result<_Tpls...>::__type
```

tuple\_cat

Definition at line 1591 of file tuple.

### 3.80.3 Variable Documentation

#### 3.80.3.1 `piecewise_construct`

`_GLIBCXX17_INLINE constexpr piecewise\_construct\_t std::piecewise_construct`

`piecewise_construct`

Definition at line 79 of file `std_pair.h`.

## 4 Namespace Documentation

### 4.1 `__gnu_cxx` Namespace Reference

#### Namespaces

- [\\_\\_detail](#)
- [typelist](#)

#### Classes

- [struct \\_\\_alloc\\_traits](#)
- [struct \\_\\_common\\_pool\\_policy](#)
- [class \\_\\_mt\\_alloc](#)
- [class \\_\\_mt\\_alloc\\_base](#)
- [struct \\_\\_per\\_type\\_pool\\_policy](#)
- [class \\_\\_pool](#)
- [class \\_\\_pool< false >](#)
- [class \\_\\_pool< true >](#)
- [class \\_\\_pool\\_alloc](#)
- [class \\_\\_pool\\_alloc\\_base](#)
- [struct \\_\\_pool\\_base](#)
- [class \\_\\_rc\\_string\\_base](#)
- [class \\_\\_scoped\\_lock](#)
- [class \\_\\_versa\\_string](#)
- [struct \\_Caster](#)
- [struct \\_Char\\_types](#)
- [class \\_ExtPtr\\_allocator](#)
- [struct \\_Invalid\\_type](#)
- [class \\_Pointer\\_adapter](#)
- [class \\_Relative\\_pointer\\_impl](#)
- [class \\_Relative\\_pointer\\_impl< const \\_Tp >](#)
- [class \\_Std\\_pointer\\_impl](#)
- [struct \\_Unqualified\\_type](#)
- [struct annotate\\_base](#)
- [class array\\_allocator](#)
- [class array\\_allocator\\_base](#)
- [class binary\\_compose](#)
- [class bitmap\\_allocator](#)
- [struct char\\_traits](#)
- [struct character](#)
- [struct condition\\_base](#)
- [struct constant\\_binary\\_fun](#)
- [struct constant\\_unary\\_fun](#)
- [struct constant\\_void\\_fun](#)
- [class debug\\_allocator](#)
- [class enc\\_filebuf](#)
- [struct encoding\\_char\\_traits](#)
- [class encoding\\_state](#)

- struct [forced\\_error](#)
- class [free\\_list](#)
- class [hash\\_map](#)
- class [hash\\_multimap](#)
- class [hash\\_multiset](#)
- class [hash\\_set](#)
- struct [limit\\_condition](#)
- class [malloc\\_allocator](#)
- class [new\\_allocator](#)
- struct [project1st](#)
- struct [project2nd](#)
- struct [random\\_condition](#)
- struct [rb\\_tree](#)
- class [recursive\\_init\\_error](#)
- class [rope](#)
- struct [select1st](#)
- struct [select2nd](#)
- class [slist](#)
- class [stdio\\_filebuf](#)
- class [stdio\\_sync\\_filebuf](#)
- class [subtractive\\_rng](#)
- struct [temporary\\_buffer](#)
- class [throw\\_allocator\\_base](#)
- struct [throw\\_allocator\\_limit](#)
- struct [throw\\_allocator\\_random](#)
- struct [throw\\_value\\_base](#)
- struct [throw\\_value\\_limit](#)
- struct [throw\\_value\\_random](#)
- class [unary\\_compose](#)

## Typedefs

- typedef void(\* **\_\_destroy\_handler**) (void \*)
- typedef [\\_\\_versa\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char >, [\\_\\_rc\\_string\\_base](#) > **\_\_rc\_string**
- typedef [\\_\\_vstring](#) **\_\_sso\_string**
- typedef [\\_\\_versa\\_string](#)< char16\_t, [std::char\\_traits](#)< char16\_t >, [std::allocator](#)< char16\_t >, [\\_\\_rc\\_string\\_base](#) > **\_\_u16rc\_string**
- typedef [\\_\\_u16vstring](#) **\_\_u16sso\_string**
- typedef [\\_\\_versa\\_string](#)< char16\_t > **\_\_u16vstring**
- typedef [\\_\\_versa\\_string](#)< char32\_t, [std::char\\_traits](#)< char32\_t >, [std::allocator](#)< char32\_t >, [\\_\\_rc\\_string\\_base](#) > **\_\_u32rc\_string**
- typedef [\\_\\_u32vstring](#) **\_\_u32sso\_string**
- typedef [\\_\\_versa\\_string](#)< char32\_t > **\_\_u32vstring**
- typedef [\\_\\_versa\\_string](#)< char > **\_\_vstring**
- typedef [\\_\\_versa\\_string](#)< wchar\_t, [std::char\\_traits](#)< wchar\_t >, [std::allocator](#)< wchar\_t >, [\\_\\_rc\\_string\\_base](#) > **\_\_wrc\_string**
- typedef [\\_\\_wvstring](#) **\_\_wsso\_string**
- typedef [\\_\\_versa\\_string](#)< wchar\_t > **\_\_wvstring**
- typedef [rope](#)< char > **crope**
- typedef [rope](#)< wchar\_t > **wrope**

## Enumerations

- enum { `_S_num_primes` }
- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

## Functions

- static void `__atomic_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- else `__atomic_add_single` (`__mem`, `__val`)
- `_Atomic_word` `__attribute` (`((__unused__))` `__exchange_and_add`(volatile `_Atomic_word` \*
- namespace `__cxx11` `__attribute` (`((__abi_tag__("cxx11")))`)
- template<class `_Tp` >  
void `__aux_require_boolean_expr` (const `_Tp` &`__t`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__const_pointer_cast` (const `_FromType` &`__arg`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__const_pointer_cast` (`_FromType` \*`__arg`)
- template<typename `_InputIterator`, typename `_Size`, typename `_OutputIterator` >  
`pair`< `_InputIterator`, `_OutputIterator` > `__copy_n` (`_InputIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`,  
[input\\_iterator\\_tag](#))
- template<typename `_RAIterator`, typename `_Size`, typename `_OutputIterator` >  
`pair`< `_RAIterator`, `_OutputIterator` > `__copy_n` (`_RAIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`,  
[random\\_access\\_iterator\\_tag](#))
- template<typename `_InputIterator`, typename `_Distance` >  
void `__distance` (`_InputIterator` `__first`, `_InputIterator` `__last`, `_Distance` &`__n`, [std::input\\_iterator\\_tag](#))
- template<typename `_RandomAccessIterator`, typename `_Distance` >  
void `__distance` (`_RandomAccessIterator` `__first`, `_RandomAccessIterator` `__last`, `_Distance` &`__n`, [std::random\\_access\\_iterator\\_tag](#))
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__dynamic_pointer_cast` (const `_FromType` &`__arg`)
- template<typename `_ToType`, typename `_FromType` >  
`_ToType` `__dynamic_pointer_cast` (`_FromType` \*`__arg`)
- void `__error_type_must_be_a_signed_integer_type` ()
- void `__error_type_must_be_an_integer_type` ()
- void `__error_type_must_be_an_unsigned_integer_type` ()
- static `_Atomic_word` `__exchange_and_add_single` (`_Atomic_word` \*`__mem`, int `__val`)
- else return `__exchange_and_add_single` (`__mem`, `__val`)
- template<class `_Concept` >  
void `__function_requires` ()
- template<typename `_Type` >  
bool `__is_null_pointer` (`_Type` \*`__ptr`)
- template<typename `_Type` >  
bool `__is_null_pointer` (`_Type`)
- bool `__is_null_pointer` (std::nullptr\_t)
- template<typename `_InputIterator1`, typename `_InputIterator2` >  
int `__lexicographical_compare_3way` (`_InputIterator1` `__first1`, `_InputIterator1` `__last1`, `_InputIterator2` `__first2`,  
`_InputIterator2` `__last2`)
- int `__lexicographical_compare_3way` (const unsigned char \*`__first1`, const unsigned char \*`__last1`, const  
unsigned char \*`__first2`, const unsigned char \*`__last2`)
- int `__lexicographical_compare_3way` (const char \*`__first1`, const char \*`__last1`, const char \*`__first2`, const  
char \*`__last2`)



- `template<typename _Tp >`  
`const _Tp & \_\_median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & \_\_median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference \_\_mutable\_reference\_at (crope &__c, size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp \_\_power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp \_\_power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`  
`_RandomAccessIterator \_\_random\_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator \_\_random\_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _ToType, typename _FromType >`  
`_ToType \_\_reinterpret\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`  
`_ToType \_\_reinterpret\_pointer\_cast (_FromType * __arg)`
- `_Slist_node_base * \_\_slist\_make\_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * \_\_slist\_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * \_\_slist\_previous (const _Slist_node_base * __head, const _Slist_node_base * __↵ node)`
- `_Slist_node_base * \_\_slist\_reverse (_Slist_node_base * __node)`
- `size_t \_\_slist\_size (_Slist_node_base * __node)`
- `void \_\_slist\_splice\_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __↵ before_last)`
- `void \_\_slist\_splice\_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<typename _ToType, typename _FromType >`  
`_ToType \_\_static\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`  
`_ToType \_\_static\_pointer\_cast (_FromType * __arg)`
- `size_t \_\_stl\_hash\_string (const char * __s)`
- `unsigned long \_\_stl\_next\_prime (unsigned long __n)`
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`  
`_Ret \_\_stoa (_TRet(* __convf)(const _CharT *, _CharT **, _Base...), const char * __name, const _CharT * __str, std::size_t * __idx, _Base... __base)`
- `void \_\_throw\_concurrency\_lock\_error ()`
- `void \_\_throw\_concurrency\_unlock\_error ()`
- `void \_\_throw\_forced\_error ()`
- `template<typename _String, typename _CharT = typename _String::value_type>`  
`_String \_\_to\_xstring (int(* __convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT * __fmt,...)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > \_\_uninitialized\_copy\_n (_InputIter __first, _Size __count, _ForwardIter __↵ result, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`  
`pair< _RandomAccessIter, _ForwardIter > \_\_uninitialized\_copy\_n (_RandomAccessIter __first, _Size __count, _ForwardIter __result, std::random\_access\_iterator\_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > \_\_uninitialized\_copy\_n (_InputIter __first, _Size __count, _ForwardIter __↵ result)`

- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a ( _InputIter __first, _Size __count, _ForwardIter __first,`  
`__result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`  
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a ( _InputIter __first, _Size __count, _ForwardIter __first,`  
`__result, std::allocator< _Tp >)`
- `void __verbose_terminate_handler ()`
- `size_t __Bit_scan_forward (size_t __num)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void __Destroy_const ( _ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void __Destroy_const ( _ForwardIterator __first, _ForwardIterator __last, allocator< _Tp >)`
- `template<class _CharT, class _Traits >`  
`void __Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`  
`bool __Rope_is_simple ( _CharT *)`
- `bool __Rope_is_simple (char *)`
- `bool __Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`  
`void __Rope_rotate ( _Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT >`  
`void __S_cond_store_eos ( _CharT &)`
- `void __S_cond_store_eos (char &__c)`
- `void __S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`  
`_CharT __S_eos ( _CharT *)`
- `template<class _CharT >`  
`bool __S_is_basic_char_type ( _CharT *)`
- `bool __S_is_basic_char_type (char *)`
- `bool __S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`  
`bool __S_is_one_byte_char_type ( _CharT *)`
- `bool __S_is_one_byte_char_type (char *)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type airy_ai ( _Tp __x)`
- `float airy_aif (float __x)`
- `long double airy_ail (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type airy_bi ( _Tp __x)`
- `float airy_bif (float __x)`
- `long double airy_bil (long double __x)`
- `template<class _Operation1, class _Operation2 >`  
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`  
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg ( _Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hypergf (float __a, float __c, float __x)`
- `long double conf_hypergl (long double __a, long double __c, long double __x)`

- `template<class _Result >`  
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`  
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`  
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator, typename _Distance >`  
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<class _Tp >`  
`_Tp identity_element (std::plus< _Tp >)`
- `template<class _Tp >`  
`_Tp identity_element (std::multiplies< _Tp >)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::*__f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::*__f)(_Arg) const)`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Tp >`  
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`  
`bool operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp, typename _Array >`  
`bool operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Alloc >`  
`bool operator!= (const debug_allocator< _Alloc > & __lhs, const debug_allocator< _Alloc > & __rhs)`
- `template<typename _Tp >`  
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > & __hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > & __hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > & __hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > & __hs2)`

- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool operator!= (const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const Pointer\_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (_Tp1 __lhs, const Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const Pointer\_adapter< _Tp1 > &__lhs, const Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool operator!= (const Pointer\_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`  
`bool operator!= (int __lhs, const Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool operator!= (const Pointer\_adapter< _Tp > &__lhs, const Pointer\_adapter< _Tp > &__rhs)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Tp, typename _Poolp >`  
`bool operator!= (const mt\_alloc< _Tp, _Poolp > &, const mt\_alloc< _Tp, _Poolp > &)`
- `template<class _Tp, class _Alloc >`  
`bool operator!= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵_IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _Tp, typename _Cond >`  
`bool operator!= (const throw\_allocator\_base< _Tp, _Cond > &, const throw\_allocator\_base< _Tp, _Cond > &)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const _CharT *__lhs, const versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const Rope\_const\_iterator< _CharT, _Alloc > &__x, const Rope\_const\_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const Rope\_iterator< _CharT, _Alloc > &__x, const Rope\_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator!= (const Rope\_char\_ptr\_proxy< _CharT, _Alloc > &__x, const Rope\_char\_ptr\_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Cond >`  
`throw\_value\_base< _Cond > operator* (const throw\_value\_base< _Cond > &__a, const throw\_value\_base< _Cond > &__b)`

- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<typename _Cond >`  
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base >  
&& __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base >  
&& __lhs, _CharT __rhs)`
- `template<class _CharT, class _Alloc >  
rope< _CharT, _Alloc > & operator+=( rope< _CharT, _Alloc > & __left, const rope< _CharT, _Alloc > & __right)`
- `template<class _CharT, class _Alloc >  
rope< _CharT, _Alloc > & operator+=( rope< _CharT, _Alloc > & __left, const _CharT * __right)`
- `template<class _CharT, class _Alloc >  
rope< _CharT, _Alloc > & operator+=( rope< _CharT, _Alloc > & __left, _CharT __right)`
- `template<class _CharT, class _Alloc >  
_Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > & __x,  
ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >  
ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > & __x, const _Rope_const_iterator< _CharT,  
_Alloc > & __y)`
- `template<class _CharT, class _Alloc >  
_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT, _Alloc > & __x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >  
ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > & __x, const _Rope_iterator< _CharT, _Alloc >  
& __y)`
- `template<typename _Cond >  
throw_value_base< _Cond > operator- (const throw_value_base< _Cond > & __a, const throw_value_base<  
_Cond > & __b)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >  
auto operator- (const __normal_iterator< _IteratorL, _Container > & __lhs, const __normal_iterator< _IteratorR,  
_Container > & __rhs) noexcept -> decltype(__lhs.base() - __rhs.base())`
- `template<typename _Iterator, typename _Container >  
__normal_iterator< _Iterator, _Container >::difference_type operator- (const __normal_iterator< _Iterator, __  
Container > & __lhs, const __normal_iterator< _Iterator, _Container > & __rhs) noexcept`
- `template<typename _Value, typename _Int, typename _St >  
bool operator< (const character< _Value, _Int, _St > & lhs, const character< _Value, _Int, _St > & rhs)`
- `template<class _CharT, class _Alloc >  
bool operator< (const _Rope_const_iterator< _CharT, _Alloc > & __x, const _Rope_const_iterator< _CharT,  
_Alloc > & __y)`
- `template<class _CharT, class _Alloc >  
bool operator< (const _Rope_iterator< _CharT, _Alloc > & __x, const _Rope_iterator< _CharT, _Alloc > & __y)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator< (const _Pointer_adapter< _Tp1 > & __lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator< ( _Tp1 __lhs, const _Pointer_adapter< _Tp2 > & __rhs)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator< (const _Pointer_adapter< _Tp1 > & __lhs, const _Pointer_adapter< _Tp2 > & __rhs)`
- `template<typename _Cond >  
bool operator< (const throw_value_base< _Cond > & __a, const throw_value_base< _Cond > & __b)`
- `template<class _Tp, class _Alloc >  
bool operator< (const slist< _Tp, _Alloc > & SL1, const slist< _Tp, _Alloc > & SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >  
bool operator< (const __normal_iterator< _IteratorL, _Container > & __lhs, const __normal_iterator< __  
IteratorR, _Container > & __rhs) noexcept`

- `template<typename _Iterator, typename _Container >`  
`bool operator< (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const ↵  
__gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1,  
__msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
__gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
_Pointer_adapter< _StoreT > &__p)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
__gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
rice_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
k_distribution< _RealType > &__x)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__o, const rope< ↵  
_CharT, _Alloc > &__r)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
__gnu_cxx::triangular_distribution< _RealType > &__x)`



- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`__gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`__gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`logistic_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`__gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`__gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const`  
`rope< _CharT, _Alloc > &__r)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Tp1 &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Tp2 &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >`  
`bool operator<= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Value, typename _Int, typename _St >`  
`bool operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<typename _Tp >`  
`bool operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`



- `template<typename _Tp >`  
`bool operator== (const malloc\_allocator< _Tp > &, const malloc\_allocator< _Tp > &)`
- `template<typename _Tp, typename _Array >`  
`bool operator== (const array\_allocator< _Tp, _Array > &, const array\_allocator< _Tp, _Array > &)`
- `template<typename _Tp >`  
`bool operator== (const \_\_pool\_alloc< _Tp > &, const \_\_pool\_alloc< _Tp > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator== (const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool operator== (const hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`  
`bool operator== (const \_\_gnu\_cxx::simd\_fast\_mersenne\_twister\_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const \_\_gnu\_cxx::simd\_fast\_mersenne\_twister\_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const \_Rope\_char\_ptr\_proxy< _CharT, _Alloc > &__x, const \_Rope\_char\_ptr\_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const \_Rope\_const\_iterator< _CharT, _Alloc > &__x, const \_Rope\_const\_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator== (const \_Rope\_iterator< _CharT, _Alloc > &__x, const \_Rope\_iterator< _CharT, _Alloc > &__y)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool operator== (const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool operator== (const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > &__lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator== (_Tp1 __lhs, const \_Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool operator== (const \_Pointer\_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`  
`bool operator== (int __lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool operator== (const \_Pointer\_adapter< _Tp > &__lhs, const \_Pointer\_adapter< _Tp > &__rhs)`
- `template<size_t _Dimen, typename _RealType >`  
`bool operator== (const \_\_gnu\_cxx::normal\_mv\_distribution< _Dimen, _RealType > &__d1, const \_\_gnu\_cxx::normal\_mv\_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _Cond >`  
`bool operator== (const throw\_value\_base< _Cond > &__a, const throw\_value\_base< _Cond > &__b)`

- `template<typename _Tp, typename _Poolp >`  
`bool operator==(const __mt_alloc<_Tp, _Poolp> &, const __mt_alloc<_Tp, _Poolp> &)`
- `template<class _Tp, class _Alloc >`  
`bool operator==(const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator==(const __normal_iterator<_IteratorL, _Container> &_lhs, const __normal_iterator<_IteratorR, _Container> &_rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool operator==(const __normal_iterator<_Iterator, _Container> &_lhs, const __normal_iterator<_Iterator, _Container> &_rhs) noexcept`
- `template<typename _Tp, typename _Cond >`  
`bool operator==(const throw_allocator_base<_Tp, _Cond> &, const throw_allocator_base<_Tp, _Cond> &)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator==(const bitmap_allocator<_Tp1> &, const bitmap_allocator<_Tp2> &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`  
`bool operator==(const __versa_string<_CharT, _Traits, _Alloc, _Base> &_lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> &_rhs)`
- `template<typename _CharT, template<typename, typename, typename> class _Base>`  
`__enable_if<std::is_char<_CharT>::value, bool>::type operator==(const __versa_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>, _Base> &_lhs, const __versa_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>, _Base> &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`  
`bool operator==(const _CharT * _lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`  
`bool operator==(const __versa_string<_CharT, _Traits, _Alloc, _Base> &_lhs, const _CharT * _rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator==(const rope<_CharT, _Alloc> &_left, const rope<_CharT, _Alloc> &_right)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>(const _Pointer_adapter<_Tp1> &_lhs, const _Pointer_adapter<_Tp2> &_rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>(const _Pointer_adapter<_Tp1> &_lhs, _Tp2 _rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>(_Tp1 _lhs, const _Pointer_adapter<_Tp2> &_rhs)`
- `template<typename _Tp >`  
`bool operator>(const _Pointer_adapter<_Tp> &_lhs, const _Pointer_adapter<_Tp> &_rhs)`
- `template<class _Tp, class _Alloc >`  
`bool operator>(const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator>(const __normal_iterator<_IteratorL, _Container> &_lhs, const __normal_iterator<_IteratorR, _Container> &_rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool operator>(const __normal_iterator<_Iterator, _Container> &_lhs, const __normal_iterator<_Iterator, _Container> &_rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`  
`bool operator>(const __versa_string<_CharT, _Traits, _Alloc, _Base> &_lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> &_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`  
`bool operator>(const __versa_string<_CharT, _Traits, _Alloc, _Base> &_lhs, const _CharT * _rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`  
`bool operator>(const _CharT * _lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> &_rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>(const _Rope_const_iterator<_CharT, _Alloc> &_x, const _Rope_const_iterator<_CharT, _Alloc> &_y)`

- `template<class _CharT, class _Alloc >`  
`bool operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp >`  
`bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >`  
`bool operator>= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵_IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _↵_CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &↵__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t ↵__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↵cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __↵msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↵cxx::beta_distribution< _RealType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↵cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, rice_↵distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, nakagami_↵distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, pareto_↵`  
`distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, k_↵`  
`distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, arcsine_↵`  
`distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, hoyt_↵`  
`distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_↵`  
`cxx::triangular_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_↵`  
`cxx::von_mises_distribution< _RealType > & __x)`
- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_↵`  
`cxx::hypergeometric_distribution< _UIntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, logistic_↵`  
`distribution< _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_↵`  
`cxx::uniform_on_sphere_distribution< _Dimen, _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_↵`  
`cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > & __x)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator ↵`  
`__out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator ↵`  
`__out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator & __rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const ↵`  
`_Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const ↵`  
`_Distance __n, _RandomNumberGenerator & __rand)`
- `void rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __↵`  
`STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> ↵`  
`__last)`
- `template<typename _Tp >`  
`void swap (_ExtPtr_allocator< _Tp > & __larg, _ExtPtr_allocator< _Tp > & __rarg)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > & __hs1, hash_set< _Val, _HashFcn, _EqualKey, ↵`  
`_Alloc > & __hs2)`

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<typename _Cond >`  
`void swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`  
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`
- `template<class _Tp, class _Alloc >`  
`void swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`
- `template<class _CharT, class __Alloc >`  
`void swap (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`
- `_Atomic_word int throw ()`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`

## Variables

- `static const _Lock_policy __default_lock_policy`
- `static _Atomic_word int __val`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

### 4.1.1 Detailed Description

GNU extensions for public use.

### 4.1.2 Function Documentation

#### 4.1.2.1 \_\_static\_pointer\_cast() [1/2]

```
template<typename _ToType, typename _FromType >
_ToType __gnu_cxx::__static_pointer_cast (
    const _FromType & __arg ) [inline]
```

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 68 of file `cast.h`.

## 4.1.2.2 \_\_static\_pointer\_cast() [2/2]

```
template<typename _ToType , typename _FromType >
_ToType __gnu_cxx::__static_pointer_cast (
    _FromType * __arg ) [inline]
```

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 96 of file `cast.h`.

## 4.1.2.3 \_Bit\_scan\_forward()

```
size_t __gnu_cxx::_Bit_scan_forward (
    size_t __num ) [inline]
```

Generic Version of the `bsf` instruction.

Definition at line 510 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

## 4.1.2.4 airy\_ai()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type __gnu_cxx::airy_ai (
    _Tp __x ) [inline]
```

Return the Airy function  $Ai(x)$  of real argument `x`.

Definition at line 1238 of file `specfun.h`.

## 4.1.2.5 airy\_aif()

```
float __gnu_cxx::airy_aif (
    float __x ) [inline]
```

Return the Airy function  $Ai(x)$  of `float` argument `x`.

Definition at line 1215 of file `specfun.h`.

#### 4.1.2.6 `airy_ail()`

```
long double __gnu_cxx::airy_ail (
    long double __x ) [inline]
```

Return the Airy function  $Ai(x)$  of long double argument x.

Definition at line 1226 of file specfun.h.

#### 4.1.2.7 `airy_bi()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type __gnu_cxx::airy_bi (
    _Tp __x ) [inline]
```

Return the Airy function  $Bi(x)$  of real argument x.

Definition at line 1273 of file specfun.h.

#### 4.1.2.8 `airy_bif()`

```
float __gnu_cxx::airy_bif (
    float __x ) [inline]
```

Return the Airy function  $Bi(x)$  of float argument x.

Definition at line 1250 of file specfun.h.

#### 4.1.2.9 `airy_bil()`

```
long double __gnu_cxx::airy_bil (
    long double __x ) [inline]
```

Return the Airy function  $Bi(x)$  of long double argument x.

Definition at line 1261 of file specfun.h.

#### 4.1.2.10 `conf_hyperg()`

```
template<typename _Tpa , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type __gnu_cxx::conf_hyperg (
    _Tpa __a,
    _Tpc __c,
    _Tp __x ) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of real numeratorial parameter a, denominatorial parameter c, and argument x.

The confluent hypergeometric function is defined by

$${}_1F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

## Parameters

$\leftrightarrow$ _a	The numeratorial parameter
$\leftrightarrow$ _c	The denominatorial parameter
$\leftrightarrow$ _x	The argument

Definition at line 1323 of file specfun.h.

## 4.1.2.11 conf\_hypergf()

```
float __gnu_cxx::conf_hypergf (
    float __a,
    float __c,
    float __x ) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of float numeratorial parameter a, denominatorial parameter c, and argument x.

## See also

conf\_hyperg for details.

Definition at line 1291 of file specfun.h.

## 4.1.2.12 conf\_hypergl()

```
long double __gnu_cxx::conf_hypergl (
    long double __a,
    long double __c,
    long double __x ) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of long double numeratorial parameter a, denominatorial parameter c, and argument x.

## See also

conf\_hyperg for details.

Definition at line 1302 of file specfun.h.



#### 4.1.2.13 `hyperg()`

```
template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type __gnu_cxx::hyperg (
    _Tpa __a,
    _Tpb __b,
    _Tpc __c,
    _Tp __x ) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of real numeratorial parameters `a` and `b`, denominatorial parameter `c`, and argument `x`.

The hypergeometric function is defined by

$${}_2F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

##### Parameters

<code>↔ _a</code>	The first numeratorial parameter
<code>↔ _b</code>	The second numeratorial parameter
<code>↔ _c</code>	The denominatorial parameter
<code>↔ _x</code>	The argument

Definition at line 1372 of file `specfun.h`.

#### 4.1.2.14 `hypergf()`

```
float __gnu_cxx::hypergf (
    float __a,
    float __b,
    float __c,
    float __x ) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of @ float numeratorial parameters `a` and `b`, denominatorial parameter `c`, and argument `x`.

##### See also

`hyperg` for details.

Definition at line 1339 of file `specfun.h`.

## 4.1.2.15 hypergl()

```
long double __gnu_cxx::hypergl (
    long double __a,
    long double __b,
    long double __c,
    long double __x ) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

Definition at line 1350 of file specfun.h.

## 4.1.2.16 operator!=(()) [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!=(
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test difference of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2388 of file vstring.h.

## 4.1.2.17 operator!=(()) [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!=(
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test difference of C string and string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2401 of file `vstring.h`.

**4.1.2.18 `operator!=()`** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test difference of string and C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2414 of file `vstring.h`.

**4.1.2.19 `operator+()`** [1/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
```

Concatenate two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 181 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.1.2.20 `operator+()` [2/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
```

Concatenate C string and string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 194 of file `vstring.tcc`.

4.1.2.21 `operator+()` [3/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    _CharT __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
```

Concatenate character and string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 211 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.1.2.22 operator+()** [4/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs )
```

Concatenate string and C string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 224 of file `vstring.tcc`.

**4.1.2.23 operator+()** [5/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    _CharT __rhs )
```

Concatenate string and character.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 241 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.1.2.24 `operator<()` [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if string precedes string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2428 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

4.1.2.25 `operator<()` [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string precedes C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2441 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.26 `operator<()`** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if C string precedes string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2454 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.27 `operator<=()`** [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if string doesn't follow string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2508 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.28 `operator<=()` [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string doesn't follow C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2521 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.29 `operator<=()` [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if C string doesn't follow string.



## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

## Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2534 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.30 `operator==()` [1/4]

```
template<typename _Tp >
bool __gnu_cxx::operator== (
    const _Pointer_adapter< _Tp > & __lhs,
    const _Pointer_adapter< _Tp > & __rhs ) [inline]
```

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

Definition at line 533 of file `pointer.h`.

4.1.2.31 `operator==()` [2/4]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator== (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test equivalence of two strings.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2337 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

#### 4.1.2.32 `operator==()` [3/4]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator== (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test equivalence of C string and string.

##### Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

##### Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2361 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

#### 4.1.2.33 `operator==()` [4/4]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator== (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test equivalence of string and C string.

##### Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

##### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2374 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

#### 4.1.2.34 `operator>()` [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if string follows string.

##### Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

##### Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2468 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

#### 4.1.2.35 `operator>()` [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string follows C string.

##### Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2481 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.36 operator>()** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if C string follows string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2494 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.37 operator>=()** [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if string doesn't precede string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2548 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.38 operator>=()** [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string doesn't precede C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2561 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.39 operator>=()** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (
    const _CharT * __lhs,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if C string doesn't precede string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2574 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.40 `swap()`**

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
void __gnu_cxx::swap (
    __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
    __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Swap contents of two strings.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2588 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

**4.2 `__gnu_cxx::__detail` Namespace Reference****Classes**

- class `__mini_vector`
- class `__Bitmap_counter`
- class `__Ffit_finder`

**Enumerations**

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

## Functions

- void `__bit_allocate` (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- void `__bit_free` (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator `__lower_bound` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
size\_t `__num_bitmaps` (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
size\_t `__num_blocks` (\_AddrPair \_\_ap)

### 4.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

### 4.2.2 Function Documentation

#### 4.2.2.1 `__bit_allocate()`

```
void __gnu_cxx::__detail::__bit_allocate (  
    size_t * __pmap,  
    size_t __pos ) throw ( )    [inline]
```

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 489 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::__M_allocate_single_object()`.

#### 4.2.2.2 `__bit_free()`

```
void __gnu_cxx::__detail::__bit_free (  
    size_t * __pmap,  
    size_t __pos ) throw ( )    [inline]
```

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 500 of file `bitmap_allocator.h`.

## 4.2.2.3 \_\_num\_bitmaps()

```
template<typename _AddrPair >
size_t __gnu_cxx::__detail::__num_bitmaps (
    _AddrPair __ap ) [inline]
```

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 277 of file bitmap\_allocator.h.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

## 4.2.2.4 \_\_num\_blocks()

```
template<typename _AddrPair >
size_t __gnu_cxx::__detail::__num_blocks (
    _AddrPair __ap ) [inline]
```

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 269 of file bitmap\_allocator.h.

Referenced by `__num_bitmaps()`.

## 4.3 \_\_gnu\_cxx::typelist Namespace Reference

## Functions

- `template<typename Fn , typename Typelist >`  
`void apply (Fn &, Typelist)`
- `template<typename Gn , typename Typelist >`  
`void apply_generator (Gn &, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV >`  
`void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist >`  
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV >`  
`void apply_generator (Fn &fn, TypelistT, TypelistV)`

## 4.3.1 Detailed Description

GNU typelist extensions for public compile-time use.



### 4.3.2 Function Documentation

#### 4.3.2.1 `apply_generator()`

```
template<typename Gn , typename Typelist >
void __gnu_cxx::typelist::apply_generator (
    Gn & ,
    Typelist )
```

Apply all typelist types to generator functor.

## 4.4 `__gnu_debug` Namespace Reference

### Classes

- class [\\_After\\_nth\\_from](#)
- struct [\\_BeforeBeginHelper](#)
- class [\\_Equal\\_to](#)
- class [\\_Not\\_equal\\_to](#)
- class [\\_Safe\\_container](#)
- class [\\_Safe\\_forward\\_list](#)
- class [\\_Safe\\_iterator](#)
- class [\\_Safe\\_iterator\\_base](#)
- class [\\_Safe\\_local\\_iterator](#)
- class [\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_Safe\\_node\\_sequence](#)
- class [\\_Safe\\_sequence](#)
- class [\\_Safe\\_sequence\\_base](#)
- class [\\_Safe\\_unordered\\_container](#)
- class [\\_Safe\\_unordered\\_container\\_base](#)
- class [\\_Safe\\_vector](#)
- struct [\\_Sequence\\_traits](#)
- class [basic\\_string](#)

### Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

## Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,  
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move` ↵  
`assign`,  
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs`, `__msg_insert_range_from` ↵  
`__self`,  
`__msg_irreflexive_ordering` }
- enum `_Distance_precision` { `__dp_none`, `__dp_equality`, `__dp_sign`, `__dp_exact` }

## Functions

- template<typename `_Iterator` >  
auto `__base` (const `std::reverse_iterator`< `_Iterator` > &`__it`) -> `decltype`(`std::make_reverse_iterator`(↵  
`__it.base`()))
- template<typename `_Iterator` >  
auto `__base` (const `std::move_iterator`< `_Iterator` > &`__it`) -> `decltype`(`std::make_move_iterator`(↵  
`__it.base`()))
- template<typename `_Iterator` >  
`_Iterator` `__base` (`_Iterator` `__it`)
- template<typename `_Iterator`, typename `_Sequence` >  
`_Iterator` `__base` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > &`__it`, `std::random_access_iterator_tag`)
- template<typename `_Iterator`, typename `_Sequence` >  
const `_Safe_iterator`< `_Iterator`, `_Sequence` > & `__base` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > &`__it`,  
`std::input_iterator_tag`)
- template<typename `_Iterator`, typename `_Sequence` >  
auto `__base` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > &`__it`) -> `decltype`(`__base`(`__it`, `std::__iterator_category`(↵  
`__it`)))
- template<typename `_Iterator` >  
bool `__check_dereferenceable` (const `_Iterator` &)
- template<typename `_Tp` >  
bool `__check_dereferenceable` (const `_Tp` \*`__ptr`)
- template<typename `_Iterator`, typename `_Sequence` >  
bool `__check_dereferenceable` (const `_Safe_local_iterator`< `_Iterator`, `_Sequence` > &`__x`)
- template<typename `_Iterator`, typename `_Sequence` >  
bool `__check_dereferenceable` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > &`__x`)
- template<typename `_ForwardIterator`, typename `_Tp` >  
bool `__check_partitioned_lower` (`_ForwardIterator` `__first`, `_ForwardIterator` `__last`, const `_Tp` &`__value`)
- template<typename `_ForwardIterator`, typename `_Tp`, typename `_Pred` >  
bool `__check_partitioned_lower` (`_ForwardIterator` `__first`, `_ForwardIterator` `__last`, const `_Tp` &`__value`, `_Pred`  
`__pred`)

- `template<typename _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _Iterator >`  
`bool __check_singular (const _Iterator &)`
- `template<typename _Tp >`  
`bool __check_singular (const _Tp * __ptr)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const _Safe_iterator_base * __x)`
- `template<typename _InputIterator >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::true\_type)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::true\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false\_type)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * __check_string (const _CharT * __s, const _Integer &__n \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _CharT >`  
`const _CharT * __check_string (const _CharT * __s)`
- `template<typename _InputIterator >`  
`_InputIterator __check_valid_range (const _InputIterator &__first, const _InputIterator &__last \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __foreign_iterator (const \_Safe\_iterator< _Iterator, _Sequence > &__it, _InputIterator __other, \_InputIterator \_\_other\_end)`
- `template<typename _Iterator, typename _Sequence, typename _Integral >`  
`bool __foreign_iterator_aux (const \_Safe\_iterator< _Iterator, _Sequence > &, _Integral, _Integral, std::true\_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __foreign_iterator_aux (const \_Safe\_iterator< _Iterator, _Sequence > &__it, _InputIterator __other, \_InputIterator \_\_other\_end, std::false\_type)`

- `template<typename _Iterator, typename _Sequence, typename _OtherIterator >`  
`bool __foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence > &__it, const __Safe_iterator< _↵`  
`OtherIterator, _Sequence > &__other, const __Safe_iterator< _OtherIterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence, typename _OtherIterator, typename _OtherSequence >`  
`bool __foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence > &__it, const __Safe_iterator< _↵`  
`OtherIterator, _OtherSequence > &, const __Safe_iterator< _OtherIterator, _OtherSequence > &)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence > &__it, const _InputIterator &__↵`  
`other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence > &__it, const _InputIterator &__↵`  
`other, const _InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence > &, const _InputIterator &, const`  
`_InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence > &__it, const typename _↵`  
`Sequence::value_type *__other)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence > &,...)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __get_distance (const std::reverse_iterator< _Iterator > &__first, const`  
`std::reverse_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __get_distance (const _Iterator &__lhs, const _Iterator &__rhs,`  
`std::random_access_iterator_tag)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __get_distance (const _Iterator &__lhs, const _Iterator &__rhs,`  
`std::input_iterator_tag)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __get_distance (const std::move_iterator< _Iterator > &__first, const`  
`std::move_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __get_distance (const _Iterator &__lhs, const _Iterator &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`std::pair< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > __get_distance`  
`(const __Safe_local_iterator< _Iterator, _Sequence > &__first, const __Safe_local_iterator< _Iterator, _Sequence`  
`> &__last, std::input_iterator_tag)`
- `template<typename _Iterator, typename _Sequence >`  
`_Distance_traits< _Iterator >::__type __get_distance (const __Safe_iterator< _Iterator, _Sequence > &__first,`  
`const __Safe_iterator< _Iterator, _Sequence > &__last, std::random_access_iterator_tag)`
- `template<typename _Iterator, typename _Sequence >`  
`_Distance_traits< _Iterator >::__type __get_distance (const __Safe_iterator< _Iterator, _Sequence > &__first,`  
`const __Safe_iterator< _Iterator, _Sequence > &__last, std::input_iterator_tag)`
- `template<typename _Iterator, typename _Sequence >`  
`_Distance_traits< _Iterator >::__type __get_distance_from_begin (const __Safe_iterator< _Iterator, _↵`  
`Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence >`  
`_Distance_traits< _Iterator >::__type __get_distance_to_end (const __Safe_iterator< _Iterator, _Sequence >`  
`&__it)`
- `template<typename _Iterator >`  
`bool __is_irreflexive (_Iterator __it)`

- `template<typename _Iterator, typename _Pred >`  
`bool __is_irreflexive_pred (_Iterator __it, _Pred __pred)`
- `template<typename _Iterator >`  
`auto __unsafe (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_iterator(__it, __unsafe(__it.base())))`
- `template<typename _Iterator >`  
`auto __unsafe (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__unsafe(__it, __unsafe(__it.base()))))`
- `template<typename _Iterator >`  
`_Iterator __unsafe (_Iterator __it)`
- `template<typename _Iterator, typename _Sequence >`  
`_Iterator __unsafe (const _Safe_local_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence >`  
`_Iterator __unsafe (const _Safe_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator >`  
`bool __valid_range (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator >`  
`bool __valid_range (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _InputIterator >`  
`bool __valid_range (const _InputIterator &__first, const _InputIterator &__last, typename _Distance_traits< _InputIterator >::__type &__dist)`
- `template<typename _InputIterator >`  
`bool __valid_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __valid_range (const _Safe_local_iterator< _Iterator, _Sequence > &__first, const _Safe_local_iterator< _Iterator, _Sequence > &__last, typename _Distance_traits< _Iterator >::__type &__dist_info)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Integral >`  
`bool __valid_range_aux (const _Integral &, const _Integral &, typename _Distance_traits< _Integral >::__type &__dist, std::__true_type)`
- `template<typename _InputIterator >`  
`bool __valid_range_aux (const _InputIterator &__first, const _InputIterator &__last, typename _Distance_traits< _InputIterator >::__type &__dist, std::__false_type)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > &getline (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > &getline (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`

- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, \_Sequence > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`\_Safe\_iterator< _Iterator, _Sequence > operator+ (typename \_Safe\_iterator< _Iterator, _Sequence >::difference_type __n, const \_Safe\_iterator< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs, const basic\_string< _CharT, \_Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const basic\_string< _CharT, _Traits, \_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic\_string< _CharT, _Traits, _Allocator > operator+ (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`\_Safe\_iterator< _IteratorL, _Sequence >::difference_type operator- (const \_Safe\_iterator< _IteratorL, \_Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`\_Safe\_iterator< _Iterator, _Sequence >::difference_type operator- (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, \_Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const \_Safe\_iterator< _Iterator, \_Sequence > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const \_Safe\_iterator< _IteratorR, \_Sequence > &__rhs) noexcept`

- Generated by Doxygen

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is,`  
`basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string< _CharT, _Traits, _Allocator > &__rhs)`

#### 4.4.1 Detailed Description

GNU debug classes for public use.

#### 4.4.2 Enumeration Type Documentation

##### 4.4.2.1 `_Distance_precision`

```
enum \_\_gnu\_debug::\_Distance\_precision
```

The precision to which we can calculate the distance between two iterators.

Definition at line 43 of file `helper_functions.h`.

#### 4.4.3 Function Documentation

##### 4.4.3.1 `__check_dereferenceable()` [1/4]

```
template<typename _Iterator >
bool \_\_gnu\_debug::\_\_check\_dereferenceable (
    const _Iterator & ) [inline]
```

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 74 of file `functions.h`.

##### 4.4.3.2 `__check_dereferenceable()` [2/4]

```
template<typename _Tp >
bool \_\_gnu\_debug::\_\_check\_dereferenceable (
    const _Tp * __ptr ) [inline]
```

Non-NULL pointers are dereferenceable.

Definition at line 80 of file `functions.h`.



#### 4.4.3.3 `__check_dereferenceable()` [3/4]

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::__check_dereferenceable (
    const _Safe_local_iterator< _Iterator, _Sequence > & __x ) [inline]
```

Safe local iterators know if they are dereferenceable.

Definition at line 436 of file `safe_local_iterator.h`.

#### 4.4.3.4 `__check_dereferenceable()` [4/4]

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::__check_dereferenceable (
    const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]
```

Safe iterators know if they are dereferenceable.

Definition at line 743 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

#### 4.4.3.5 `__check_singular()`

```
template<typename _Tp >
bool __gnu_debug::__check_singular (
    const _Tp * __ptr ) [inline]
```

Non-NULL pointers are nonsingular.

Definition at line 67 of file `functions.h`.

#### 4.4.3.6 `__check_singular_aux()`

```
bool __gnu_debug::__check_singular_aux (
    const _Safe_iterator_base * __x ) [inline]
```

Iterators that derive from `_Safe_iterator_base` can be determined singular or non-singular.

Definition at line 168 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

## 4.4.3.7 \_\_check\_string() [1/2]

```
template<typename _CharT , typename _Integer >
const _CharT* __gnu_debug::__check_string (
    const _CharT * __s,
    const _Integer &__n __attribute__((unused)) ) [inline]
```

Checks that \_\_s is non-NULL or \_\_n == 0, and then returns \_\_s.

Definition at line 217 of file functions.h.

## 4.4.3.8 \_\_check\_string() [2/2]

```
template<typename _CharT >
const _CharT* __gnu_debug::__check_string (
    const _CharT * __s ) [inline]
```

Checks that \_\_s is non-NULL and then returns \_\_s.

Definition at line 229 of file functions.h.

## 4.4.3.9 \_\_foreign\_iterator\_aux2() [1/2]

```
template<typename _Iterator , typename _Sequence , typename _OtherIterator >
bool __gnu_debug::__foreign_iterator_aux2 (
    const _Safe_iterator< _Iterator, _Sequence > & __it,
    const _Safe_iterator< _OtherIterator, _Sequence > & __other,
    const _Safe_iterator< _OtherIterator, _Sequence > & ) [inline]
```

Handle debug iterators from the same type of container.

Definition at line 150 of file functions.h.

## 4.4.3.10 \_\_foreign\_iterator\_aux2() [2/2]

```
template<typename _Iterator , typename _Sequence , typename _OtherIterator , typename _OtherSequence >
bool __gnu_debug::__foreign_iterator_aux2 (
    const _Safe_iterator< _Iterator, _Sequence > & __it,
    const _Safe_iterator< _OtherIterator, _OtherSequence > & ,
    const _Safe_iterator< _OtherIterator, _OtherSequence > & ) [inline]
```

Handle debug iterators from different types of container.

Definition at line 159 of file functions.h.

**4.4.3.11** `__get_distance()` [1/3]

```
template<typename _Iterator >
_Distance_traits<_Iterator>::__type __gnu_debug::__get_distance (
    const _Iterator & __lhs,
    const _Iterator & __rhs,
    std::random_access_iterator_tag ) [inline]
```

Determine the distance between two iterators with some known precision.

Definition at line 83 of file `helper_functions.h`.

References `std::make_pair()`.

Referenced by `__valid_range_aux()`.

**4.4.3.12** `__get_distance()` [2/3]

```
template<typename _Iterator , typename _Sequence >
std::pair<typename std::iterator_traits<_Iterator>::difference_type, _Distance_precision> __↔
gnu_debug::__get_distance (
    const _Safe_local_iterator< _Iterator, _Sequence > & __first,
    const _Safe_local_iterator< _Iterator, _Sequence > & __last,
    std::input_iterator_tag ) [inline]
```

Safe local iterators need a special method to get distance between each other.

Definition at line 453 of file `safe_local_iterator.h`.

**4.4.3.13** `__get_distance()` [3/3]

```
template<typename _Iterator , typename _Sequence >
_Distance_traits<_Iterator>::__type __gnu_debug::__get_distance (
    const _Safe_iterator< _Iterator, _Sequence > & __first,
    const _Safe_iterator< _Iterator, _Sequence > & __last,
    std::random_access_iterator_tag ) [inline]
```

Safe iterators can help to get better distance knowledge.

Definition at line 757 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base()`, and `std::make_pair()`.

4.4.3.14 `__valid_range()` [1/3]

```
template<typename _InputIterator >
bool __gnu_debug::__valid_range (
    const _InputIterator & __first,
    const _InputIterator & __last,
    typename _Distance_traits< _InputIterator >::__type & __dist ) [inline]
```

Don't know what these iterators are, or if they are even iterators (we may get an integral type for InputIterator), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 152 of file `helper_functions.h`.

References `__valid_range_aux()`.

4.4.3.15 `__valid_range()` [2/3]

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::__valid_range (
    const _Safe_local_iterator< _Iterator, _Sequence > & __first,
    const _Safe_local_iterator< _Iterator, _Sequence > & __last,
    typename _Distance_traits< _Iterator >::__type & __dist_info ) [inline]
```

Safe local iterators know how to check if they form a valid range.

Definition at line 443 of file `safe_local_iterator.h`.

4.4.3.16 `__valid_range()` [3/3]

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::__valid_range (
    const _Safe_iterator< _Iterator, _Sequence > & __first,
    const _Safe_iterator< _Iterator, _Sequence > & __last,
    typename _Distance_traits< _Iterator >::__type & __dist ) [inline]
```

Safe iterators know how to check if they form a valid range.

Definition at line 749 of file `safe_iterator.h`.

#### 4.4.3.17 `__valid_range_aux()` [1/2]

```
template<typename _Integral >
bool __gnu_debug::__valid_range_aux (
    const _Integral & ,
    const _Integral & ,
    typename _Distance_traits< _Integral >::__type & __dist,
    std::__true_type ) [inline]
```

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 109 of file `helper_functions.h`.

References `std::make_pair()`.

Referenced by `__valid_range()`.

#### 4.4.3.18 `__valid_range_aux()` [2/2]

```
template<typename _InputIterator >
bool __gnu_debug::__valid_range_aux (
    const _InputIterator & __first,
    const _InputIterator & __last,
    typename _Distance_traits< _InputIterator >::__type & __dist,
    std::__false_type ) [inline]
```

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 122 of file `helper_functions.h`.

References `__get_distance()`, `std::pair< _T1, _T2 >::first`, and `std::pair< _T1, _T2 >::second`.

## 4.5 `__gnu_internal` Namespace Reference

### 4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

4.6 `__gnu_parallel` Namespace Reference

## Classes

- struct [\\_\\_accumulate\\_binop\\_reduct](#)
- struct [\\_\\_accumulate\\_selector](#)
- struct [\\_\\_adjacent\\_difference\\_selector](#)
- struct [\\_\\_adjacent\\_find\\_selector](#)
- class [\\_\\_binder1st](#)
- class [\\_\\_binder2nd](#)
- struct [\\_\\_count\\_if\\_selector](#)
- struct [\\_\\_count\\_selector](#)
- struct [\\_\\_fill\\_selector](#)
- struct [\\_\\_find\\_first\\_of\\_selector](#)
- struct [\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_for\\_each\\_selector](#)
- struct [\\_\\_generate\\_selector](#)
- struct [\\_\\_generic\\_find\\_selector](#)
- struct [\\_\\_generic\\_for\\_each\\_selector](#)
- struct [\\_\\_identity\\_selector](#)
- struct [\\_\\_inner\\_product\\_selector](#)
- struct [\\_\\_max\\_element\\_reduct](#)
- struct [\\_\\_min\\_element\\_reduct](#)
- struct [\\_\\_mismatch\\_selector](#)
- struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_replace\\_if\\_selector](#)
- struct [\\_\\_replace\\_selector](#)
- struct [\\_\\_transform1\\_selector](#)
- struct [\\_\\_transform2\\_selector](#)
- class [\\_\\_unary\\_negate](#)
- struct [\\_DRandomShufflingGlobalData](#)
- struct [\\_DRSSorterPU](#)
- struct [\\_DummyReduct](#)
- class [\\_EqualFromLess](#)
- struct [\\_EqualTo](#)
- class [\\_GuardedIterator](#)
- class [\\_IteratorPair](#)
- class [\\_IteratorTriple](#)
- struct [\\_Job](#)
- struct [\\_Less](#)
- class [\\_Lexicographic](#)
- class [\\_LexicographicReverse](#)
- class [\\_LoserTree](#)
- class [\\_LoserTree< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeBase](#)

- class [\\_LoserTreePointer](#)
- class [\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerBase](#)
- class [\\_LoserTreePointerUnguarded](#)
- class [\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerUnguardedBase](#)
- struct [\\_LoserTreeTraits](#)
- class [\\_LoserTreeUnguarded](#)
- class [\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeUnguardedBase](#)
- struct [\\_Multiplies](#)
- struct [\\_Nothing](#)
- struct [\\_Piece](#)
- struct [\\_Plus](#)
- struct [\\_PMWMSSortingData](#)
- class [\\_PseudoSequence](#)
- class [\\_PseudoSequenceIterator](#)
- struct [\\_QSBThreadLocal](#)
- class [\\_RandomNumber](#)
- class [\\_RestrictedBoundedConcurrentQueue](#)
- struct [\\_SamplingSorter](#)
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)
- struct [\\_Settings](#)
- struct [\\_SplitConsistently](#)
- struct [\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [balanced\\_quicksort\\_tag](#)
- struct [balanced\\_tag](#)
- struct [constant\\_size\\_blocks\\_tag](#)
- struct [default\\_parallel\\_tag](#)
- struct [equal\\_split\\_tag](#)
- struct [exact\\_tag](#)
- struct [find\\_tag](#)
- struct [growing\\_blocks\\_tag](#)
- struct [multiway\\_mergesort\\_exact\\_tag](#)
- struct [multiway\\_mergesort\\_sampling\\_tag](#)
- struct [multiway\\_mergesort\\_tag](#)
- struct [omp\\_loop\\_static\\_tag](#)
- struct [omp\\_loop\\_tag](#)
- struct [parallel\\_tag](#)
- struct [quicksort\\_tag](#)
- struct [sampling\\_tag](#)
- struct [sequential\\_tag](#)
- struct [unbalanced\\_tag](#)

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum `_AlgorithmStrategy` { `heuristic`, `force_sequential`, `force_parallel` }
- enum `_FindAlgorithm` { `GROWING_BLOCKS`, `CONSTANT_SIZE_BLOCKS`, `EQUAL_SPLIT` }
- enum `_MultiwayMergeAlgorithm` { `LOSER_TREE` }
- enum `_Parallelism` { `sequential`, `parallel_unbalanced`, `parallel_balanced`, `parallel_omp_loop`, `parallel_omp_loop_static`, `parallel_taskqueue` }
- enum `_PartialSumAlgorithm` { `RECURSIVE`, `LINEAR` }
- enum `_SortAlgorithm` { `MWMS`, `QS`, `QS_BALANCED` }
- enum `_SplittingAlgorithm` { `SAMPLING`, `EXACT` }

## Functions

- template<typename `_Tp` >  
`_Tp __add_omp` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- template<typename `_RAlter`, typename `_DifferenceTp` >  
void `__calc_borders` (`_RAlter` `__elements`, `_DifferenceTp` `__length`, `_DifferenceTp *``__off`)
- template<typename `_Tp` >  
bool `__cas_omp` (volatile `_Tp *``__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- template<typename `_Tp` >  
bool `__compare_and_swap` (volatile `_Tp *``__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- template<typename `_Iter`, typename `_OutputIterator` >  
`_OutputIterator` `__copy_tail` (std::pair< `_Iter`, `_Iter` > `__b`, std::pair< `_Iter`, `_Iter` > `__e`, `_OutputIterator` `__r`)
- void `__decode2` (`_CASable` `__x`, int &`__a`, int &`__b`)
- template<typename `_RAlter`, typename `_DifferenceTp` >  
void `__determine_samples` (`_PMWMSortingData`< `_RAlter` > `*__sd`, `_DifferenceTp` `__num_samples`)
- `_CASable` `__encode2` (int `__a`, int `__b`)
- template<typename `_DifferenceType`, typename `_OutputIterator` >  
`_OutputIterator` `__equally_split` (`_DifferenceType` `__n`, `_ThreadIndex` `__num_threads`, `_OutputIterator` `__s`)
- template<typename `_DifferenceType` >  
`_DifferenceType` `__equally_split_point` (`_DifferenceType` `__n`, `_ThreadIndex` `__num_threads`, `_ThreadIndex` `__thread_no`)
- template<typename `_Tp` >  
`_Tp` `__fetch_and_add` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- template<typename `_RAlter1`, typename `_RAlter2`, typename `_Pred`, typename `_Selector` >  
std::pair< `_RAlter1`, `_RAlter2` > `__find_template` (`_RAlter1` `__begin1`, `_RAlter1` `__end1`, `_RAlter2` `__begin2`, `_Pred` `__pred`, `_Selector` `__selector`)
- template<typename `_RAlter1`, typename `_RAlter2`, typename `_Pred`, typename `_Selector` >  
std::pair< `_RAlter1`, `_RAlter2` > `__find_template` (`_RAlter1` `__begin1`, `_RAlter1` `__end1`, `_RAlter2` `__begin2`, `_Pred` `__pred`, `_Selector` `__selector`, `equal_split_tag`)
- template<typename `_RAlter1`, typename `_RAlter2`, typename `_Pred`, typename `_Selector` >  
std::pair< `_RAlter1`, `_RAlter2` > `__find_template` (`_RAlter1` `__begin1`, `_RAlter1` `__end1`, `_RAlter2` `__begin2`, `_Pred` `__pred`, `_Selector` `__selector`, `growing_blocks_tag`)
- template<typename `_RAlter1`, typename `_RAlter2`, typename `_Pred`, typename `_Selector` >  
std::pair< `_RAlter1`, `_RAlter2` > `__find_template` (`_RAlter1` `__begin1`, `_RAlter1` `__end1`, `_RAlter2` `__begin2`, `_Pred` `__pred`, `_Selector` `__selector`, `constant_size_blocks_tag`)
- template<typename `_Iter`, typename `_UserOp`, typename `_Functionality`, typename `_Red`, typename `_Result` >  
`_UserOp` `__for_each_template_random_access` (`_Iter` `__begin`, `_Iter` `__end`, `_UserOp` `__user_op`, `_Functionality` &`__functionality`, `_Red` `__reduction`, `_Result` `__reduction_start`, `_Result` &`__output`, typename std::iterator\_traits< `_Iter` >::difference\_type `__bound`, `_Parallelism` `__parallelism_tag`)



- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_ed ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,`  
`_Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_omp_loop ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,`  
`_Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_omp_loop_static ( _RAIter __begin, _RAIter __end, _Op __o, _Fu`  
`& __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type`  
`__bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_workstealing ( _RAIter __begin, _RAIter __end, _Op __op, _Fu & __f,`  
`_Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type`  
`__bound)`
- `_ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const _Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool __is_sorted ( _Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __median_of_three_iterators ( _RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __`  
`end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2`  
`__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_usual ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2`  
`__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __`  
`end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare`  
`__comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1 __`  
`end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare`  
`__comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation`  
`__bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_basecase ( _Iter __begin, _Iter __end, _OutputIterator __result, __`  
`BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_linear ( _Iter __begin, _Iter __end, _OutputIterator __result, _Binary`  
`Operation __bin_op, typename std::iterator_traits< _Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_partition ( _RAIter __begin, _RAIter __end, _Predicate`  
`__pred, _ThreadIndex __num_threads)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=`  
`__RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::`  
`difference_type __n, __ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator __parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _`  
`OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __`  
`end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _Output`  
`Iterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag`  
`__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, __ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, __ThreadIndex __num`  
`threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _`  
`Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::`  
`iterator_traits< _RAIter >::difference_type __num_samples, __ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, __ThreadIndex __num_threads)`

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator \_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator \_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _RAlter, typename _Compare >`  
`void \_\_qsb\_conquer (_QSBThreadLocal< _RAlter > ** __tls, _RAlter __begin, _RAlter __end, _Compare __comp,`  
`\_\_ThreadIndex __iam, \_\_ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAlter, typename _Compare >`  
`std::iterator_traits< _RAlter >::difference_type \_\_qsb\_divide (_RAlter __begin, _RAlter __end, _Compare __comp,`  
`\_\_ThreadIndex __num_threads)`
- `template<typename _RAlter, typename _Compare >`  
`void \_\_qsb\_local\_sort\_with\_helping (_QSBThreadLocal< _RAlter > ** __tls, _Compare & __comp, \_\_ThreadIndex`  
`__iam, bool __wait)`
- `template<typename _RandomNumberGenerator >`  
`int \_\_random\_number\_pow2 (int __logp, _RandomNumberGenerator & __rng)`
- `template<typename _Size >`  
`_Size \_\_rd\_log2 (_Size __n)`
- `template<typename _Tp >`  
`_Tp \_\_round\_up\_to\_pow2 (_Tp __x)`
- `template<typename __RAlter1, typename __RAlter2, typename _Pred >`  
`__RAlter1 \_\_search\_template (__RAlter1 __begin1, __RAlter1 __end1, __RAlter2 __begin2, __RAlter2 __end2,`  
`__Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare`  
`>`  
`_RAlter3 \_\_sequential\_multiway\_merge (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3`  
`__target, const typename std::iterator_traits< typename std::iterator_traits< _RAlterIterator >::value_type &`  
`::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void \_\_sequential\_random\_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator & __rng)`
- `template<typename _Iter >`  
`void \_\_shrink (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length)`
- `template<typename _Iter >`  
`void \_\_shrink\_and\_double (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length,`  
`const bool __make_twice)`
- `void \_\_yield ()`
- `template<typename _Iter, typename _FuncType >`  
`size_t \_\_list\_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts,`  
`_FuncType & __f, int __oversampling=0)`
- `template<typename _Tp >`  
`const _Tp & max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`  
`void multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets,`  
`_Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type >::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType __offset,`  
`_Compare __comp=std::less< _Tp >())`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway\_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target,`  
`_DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←`  
`__target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←`  
`__target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←`  
`__target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <←`  
`__target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp`  
`, typename _Compare >`  
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 <←`  
`target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp`  
`, typename _Compare >`  
`_RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 <←`  
`target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _Difference<←`  
`Type __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, <←`  
`DifferenceType > > *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 <←`  
`__target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare`  
`>`  
`_RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _R<←`  
`Alter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value<←`  
`type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, <←`  
`_RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value<←`  
`_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, <←`  
`DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< <←`  
`DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _R<←`  
`AlterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <←`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <←`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <←`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter,`  
`typename _Compare >`  
`_RAIter3 parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 ↵`  
`target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void parallel\_sort\_mwms\_pu (\_PMWMSortingData< _RAIter > *__sd, _Compare & __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_CASable\\_mask](#)

### 4.6.1 Detailed Description

GNU parallel code for public use.

## 4.6.2 Typedef Documentation

### 4.6.2.1 `_BinIndex`

```
typedef unsigned short __gnu_parallel::_BinIndex
```

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

### 4.6.2.2 `_CASable`

```
typedef int64_t __gnu_parallel::_CASable
```

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

### 4.6.2.3 `_SequenceIndex`

```
typedef uint64_t __gnu_parallel::_SequenceIndex
```

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

### 4.6.2.4 `_ThreadIndex`

```
typedef uint16_t __gnu_parallel::_ThreadIndex
```

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

## 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 `_AlgorithmStrategy`

```
enum __gnu_parallel::_AlgorithmStrategy
```

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

#### 4.6.3.2 `_FindAlgorithm`

```
enum __gnu_parallel::_FindAlgorithm
```

Find algorithms:

Definition at line 106 of file types.h.

#### 4.6.3.3 `_MultiwayMergeAlgorithm`

```
enum __gnu_parallel::_MultiwayMergeAlgorithm
```

Merging algorithms:

Definition at line 85 of file types.h.

#### 4.6.3.4 `_Parallelism`

```
enum __gnu_parallel::_Parallelism
```

Run-time equivalents for the compile-time tags.

Enumerator

<code>sequential</code>	Not parallel.
<code>parallel_unbalanced</code>	Parallel unbalanced (equal-sized chunks).
<code>parallel_balanced</code>	Parallel balanced (work-stealing).
<code>parallel_omp_loop</code>	Parallel with OpenMP dynamic load-balancing.
<code>parallel_omp_loop_static</code>	Parallel with OpenMP static load-balancing.
<code>parallel_taskqueue</code>	Parallel with OpenMP taskqueue construct.

Definition at line 44 of file types.h.

#### 4.6.3.5 `_PartialSumAlgorithm`

```
enum __gnu_parallel::_PartialSumAlgorithm
```

Partial sum algorithms: recursive, linear.

Definition at line 91 of file `types.h`.

#### 4.6.3.6 `_SortAlgorithm`

```
enum __gnu_parallel::_SortAlgorithm
```

Sorting algorithms:

Definition at line 76 of file `types.h`.

#### 4.6.3.7 `_SplittingAlgorithm`

```
enum __gnu_parallel::_SplittingAlgorithm
```

Sorting/merging algorithms: sampling, `__exact`.

Definition at line 98 of file `types.h`.

### 4.6.4 Function Documentation

#### 4.6.4.1 `__calc_borders()`

```
template<typename _RAIter , typename _DifferenceTp >
void __gnu_parallel::__calc_borders (
    _RAIter __elements,
    _DifferenceTp __length,
    _DifferenceTp * __off )
```

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

##### Parameters

<code>__elements</code>	Begin iterator of sequence to search for.
<code>__length</code>	Length of sequence to search for.
<code>__off</code>	Returned <code>__offsets</code> .



Definition at line 51 of file search.h.

Referenced by `__search_template()`.

#### 4.6.4.2 `__compare_and_swap()`

```
template<typename _Tp >
bool __gnu_parallel::__compare_and_swap (
    volatile _Tp * __ptr,
    _Tp __comparand,
    _Tp __replacement ) [inline]
```

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

##### Parameters

<code>__ptr</code>	Pointer to signed integer.
<code>__comparand</code>	Compare value.
<code>__replacement</code>	Replacement value.

Definition at line 108 of file parallel/compatibility.h.

Referenced by `__parallel_partition()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > ::pop_back()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > > ::pop_↵front()`.

#### 4.6.4.3 `__decode2()`

```
void __gnu_parallel::__decode2 (
    _CASable __x,
    int & __a,
    int & __b ) [inline]
```

Decode two integers from one `gnu_parallel::_CASable`.

##### Parameters

<code>↵ __x</code>	<code>__gnu_parallel::_CASable</code> to decode integers from.
<code>↵ __a</code>	First integer, to be decoded from the most-significant <code>_CASable_bits/2</code> bits of <code>__x</code> .
<code>↵ __b</code>	Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits of <code>__x</code> .

See also

[\\_\\_encode2](#)

Definition at line 133 of file parallel/base.h.

References [\\_CASable\\_bits](#), and [\\_CASable\\_mask](#).

Referenced by [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< pair< \\_RAIter, \\_RAIter > >::pop\\_back\(\)](#), [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< pair< \\_RAIter, \\_RAIter > >::pop\\_front\(\)](#), and [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< pair< \\_RAIter, \\_RAIter > >::push\\_front\(\)](#).

#### 4.6.4.4 \_\_determine\_samples()

```
template<typename _RAIter , typename _DifferenceTp >
void __gnu_parallel::__determine_samples (
    _PMWMSSortingData< _RAIter > * __sd,
    _DifferenceTp __num_samples )
```

Select [\\_M\\_samples](#) from a sequence.

Parameters

<a href="#">__sd</a>	Pointer to algorithm data. Result will be placed in <a href="#">__sd-&gt;_M_samples</a> .
<a href="#">__num_samples</a>	Number of <a href="#">_M_samples</a> to select.

Definition at line 97 of file multiway\_mergesort.h.

References [\\_\\_equally\\_split\(\)](#), [\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >::\\_M\\_samples](#), [\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >::\\_M\\_source](#), and [\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >::\\_M\\_starts](#).

#### 4.6.4.5 \_\_encode2()

```
_CASable __gnu_parallel::__encode2 (
    int __a,
    int __b ) [inline]
```

Encode two integers into one [gnu\\_parallel::\\_CASable](#).

Parameters

<a href="#">__a</a>	First integer, to be encoded in the most-significant <a href="#">_CASable_bits/2</a> bits.
<a href="#">__b</a>	Second integer, to be encoded in the least-significant <a href="#">_CASable_bits/2</a> bits.

**Returns**

value encoding `__a` and `__b`.

**See also**

`__decode2`

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::_pop_back()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::_pop_front()`.

**4.6.4.6 `__equally_split()`**

```
template<typename _DifferenceType , typename _OutputIterator >
_OutputIterator __gnu_parallel::__equally_split (
    _DifferenceType __n,
    \_ThreadIndex __num_threads,
    _OutputIterator __s )
```

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

**Parameters**

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__s</code>	Splitters

**Returns**

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, and `__search_template()`.

4.6.4.7 `__equally_split_point()`

```
template<typename _DifferenceType >
_DifferenceType __gnu_parallel::__equally_split_point (
    _DifferenceType __n,
    _ThreadIndex __num_threads,
    _ThreadIndex __thread_no )
```

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

## Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__thread_no</code>	Number of threads

## Returns

splitting point

Definition at line 75 of file `equally_split.h`.

4.6.4.8 `__fetch_and_add()`

```
template<typename _Tp >
_Tp __gnu_parallel::__fetch_and_add (
    volatile _Tp * __ptr,
    _Tp __addend ) [inline]
```

Add a value to a variable, atomically.

## Parameters

<code>__ptr</code>	Pointer to a signed integer.
<code>__addend</code>	Value to add.

Definition at line 74 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`.

4.6.4.9 `__find_template()` [1/4]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector ) [inline]
```

Parallel `std::find`, switch for different algorithms.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

Definition at line 60 of file `find.h`.

References `__gnu_parallel::_Settings::get()`.

4.6.4.10 `__find_template()` [2/4]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector,
    equal_split_tag )
```

Parallel `std::find`, equal splitting variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

**Returns**

Place of finding in both sequences.

Definition at line 97 of file find.h.

References `__equally_split()`, and `_GLIBCXX_CALL`.

**4.6.4.11 \_\_find\_template()** [3/4]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector,
    growing_blocks_tag )
```

Parallel `std::find`, growing block size variant.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

**Returns**

Place of finding in both sequences.

**See also**

`__gnu_parallel::_Settings::find_sequential_search_size`  
`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_sequential_search_size`, `std::pair<_T1, _T2>::first`, and `__gnu_parallel::_Settings::get()`.

4.6.4.12 `__find_template()` [ 4 / 4 ]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred,
    _Selector __selector,
    constant_size_blocks_tag )
```

Parallel `std::find`, constant block size variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

## Returns

Place of finding in both sequences.

## See also

`__gnu_parallel::_Settings::find_sequential_search_size`

`__gnu_parallel::_Settings::find_block_size` There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, `std::pair<_T1, _T2>::first`, and `__gnu_parallel::_Settings::get()`.

4.6.4.13 `__for_each_template_random_access()`

```
template<typename _IIter , typename _UserOp , typename _Functionality , typename _Red , typename
_Result >
_UserOp __gnu_parallel::__for_each_template_random_access (
    _IIter __begin,
    _IIter __end,
    _UserOp __user_op,
    _Functionality & __functionality,
```

```
    _Red __reduction,  
    _Result __reduction_start,  
    _Result & __output,  
    typename std::iterator_traits< _Iter >::difference_type __bound,  
    _Parallelism __parallelism_tag )
```

Chose the desired algorithm by evaluating `__parallelism_tag`.



## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__user_op</code>	A user-specified functor (comparator, predicate, associative operator,...)
<code>__functionality</code>	functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. accumulate, <code>for_each</code> ,...)
<code>__reduction</code>	Reduction functor.
<code>__reduction_start</code>	Initial value for reduction.
<code>__output</code>	Output iterator.
<code>__bound</code>	Maximum number of elements processed.
<code>__parallelism_tag</code>	Parallelization method

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

4.6.4.14 `__for_each_template_random_access_ed()`

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_ed (
    _RAIter __begin,
    _RAIter __end,
    _Op __o,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound )
```

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...)
<code>__f</code>	Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file par\_loop.h.

Referenced by \_\_for\_each\_template\_random\_access().

**4.6.4.15 \_\_for\_each\_template\_random\_access\_omp\_loop()**

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (
    _RAIter __begin,
    _RAIter __end,
    _Op __o,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound )
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

**Parameters**

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, etc.).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file omp\_loop.h.

Referenced by \_\_for\_each\_template\_random\_access().

#### 4.6.4.16 `__for_each_template_random_access_omp_loop_static()`

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (
    _RAIter __begin,
    _RAIter __end,
    _Op __o,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound )
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

##### Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

##### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

#### 4.6.4.17 `__for_each_template_random_access_workstealing()`

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_workstealing (
    _RAIter __begin,
    _RAIter __end,
    _Op __op,
    _Fu & __f,
    _Red __r,
    _Result __base,
    _Result & __output,
    typename std::iterator_traits< _RAIter >::difference_type __bound )
```

Work stealing algorithm for random access iterators.

Uses  $O(1)$  additional memory. Synchronization at job lists is done with atomic operations.

## Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__op</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

Referenced by `__for_each_template_random_access()`.

4.6.4.18 `__is_sorted()`

```
template<typename _IIter , typename _Compare >
bool __gnu_parallel::__is_sorted (
    _IIter __begin,
    _IIter __end,
    _Compare __comp )
```

Check whether `[__begin, __end)` is sorted according to `__comp`.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.

## Returns

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

4.6.4.19 `__median_of_three_iterators()`

```
template<typename _RAIter , typename _Compare >
_RAIter __gnu_parallel::__median_of_three_iterators (
    _RAIter __a,
    _RAIter __b,
    _RAIter __c,
    _Compare __comp )
```

Compute the median of three referenced elements, according to `__comp`.

## Parameters

<code>__a</code>	First iterator.
<code>__b</code>	Second iterator.
<code>__c</code>	Third iterator.
<code>__comp</code>	Comparator.

Definition at line 398 of file `parallel/base.h`.

4.6.4.20 `__merge_advance()`

```
template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _Difference←
Tp , typename _Compare >
_OutputIterator __gnu_parallel::__merge_advance (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _OutputIterator __target,
    _DifferenceTp __max_length,
    _Compare __comp ) [inline]
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

**Returns**

Output end iterator.

Definition at line 171 of file `merge.h`.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.21 `__merge_advance_movc()`**

```
template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceType ,
typename _Compare >
_OutputIterator __gnu_parallel::__merge_advance_movc (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _OutputIterator __target,
    _DifferenceType __max_length,
    _Compare __comp )
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

**Parameters**

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

**Returns**

Output end iterator.

Definition at line 105 of file `merge.h`.

Referenced by `__merge_advance()`.

#### 4.6.4.22 `__merge_advance_usual()`

```
template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceType ,
typename _Compare >
_OutputIterator __gnu_parallel::__merge_advance_usual (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _OutputIterator __target,
    _DifferenceType __max_length,
    _Compare __comp )
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

##### Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

##### Returns

Output end iterator.

Definition at line 57 of file `merge.h`.

#### 4.6.4.23 `__parallel_merge_advance()` [1/2]

```
template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare >
_RAIter3 __gnu_parallel::__parallel_merge_advance (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter2 & __begin2,
    _RAIter2 __end2,
    _RAIter3 __target,
    typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
    _Compare __comp ) [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

## Returns

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

## 4.6.4.24 \_\_parallel\_merge\_advance() [2/2]

```
template<typename _RAIter1 , typename _RAIter3 , typename _Compare >
_RAIter3 __gnu_parallel::__parallel_merge_advance (
    _RAIter1 & __begin1,
    _RAIter1 __end1,
    _RAIter1 & __begin2,
    _RAIter1 __end2,
    _RAIter3 __target,
    typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
    _Compare __comp ) [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.



**Returns**

Output end iterator.

Definition at line 223 of file merge.h.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**4.6.4.25 `__parallel_nth_element()`**

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_nth_element (
    _RAIter __begin,
    _RAIter __nth,
    _RAIter __end,
    _Compare __comp )
```

Parallel implementation of `std::nth_element()`.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__nth</code>	Iterator of element that must be in position afterwards.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

Definition at line 332 of file partition.h.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `std::max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, and `__gnu_parallel::_Settings::partition_minimal_n`.

Referenced by `__parallel_partial_sort()`.

**4.6.4.26 `__parallel_partial_sort()`**

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_partial_sort (
    _RAIter __begin,
    _RAIter __middle,
    _RAIter __end,
    _Compare __comp )
```

Parallel implementation of `std::partial_sort()`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__middle</code>	Sort until this position.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

## 4.6.4.27 \_\_parallel\_partial\_sum()

```
template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum (
    _IIter __begin,
    _IIter __end,
    _OutputIterator __result,
    _BinaryOperation __bin_op )
```

Parallel partial sum front-`__end`.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.

## Returns

End iterator of output sequence.

Definition at line 205 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

## 4.6.4.28 \_\_parallel\_partial\_sum\_basecase()

```
template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (
    _IIter __begin,
    _IIter __end,
    _OutputIterator __result,
    _BinaryOperation __bin_op,
    typename std::iterator_traits< _IIter >::value_type __value )
```

Base case prefix sum routine.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__value</code>	Start value. Must be passed since the neutral element is unknown in general.

**Returns**

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

**4.6.4.29 `__parallel_partial_sum_linear()`**

```
template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (
    _IIter __begin,
    _IIter __end,
    _OutputIterator __result,
    _BinaryOperation __bin_op,
    typename std::iterator_traits< _IIter >::difference_type __n )
```

Parallel partial sum implementation, two-phase approach, no recursion.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__n</code>	Length of sequence.

**Returns**

End iterator of output sequence.

Definition at line 89 of file `partial_sum.h`.

References `__equally_split()`, `__parallel_partial_sum_basecase()`, `std::accumulate()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

## 4.6.4.30 \_\_parallel\_partition()

```
template<typename _RAIter , typename _Predicate >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_partition (
    _RAIter __begin,
    _RAIter __end,
    _Predicate __pred,
    _ThreadIndex __num_threads )
```

Parallel implementation of std::partition.

## Parameters

<code>__begin</code>	Begin iterator of input sequence to split.
<code>__end</code>	End iterator of input sequence to split.
<code>__pred</code>	Partition predicate, possibly including some kind of pivot.
<code>__num_threads</code>	Maximum number of threads to use for this task.

## Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file partition.h.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::partition_chunk_share`, and `__gnu_parallel::Settings::partition_chunk_size`.

Referenced by `__parallel_nth_element()`.

## 4.6.4.31 \_\_parallel\_random\_shuffle()

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__parallel_random_shuffle (
    _RAIter __begin,
    _RAIter __end,
    _RandomNumberGenerator __rng = _RandomNumber() ) [inline]
```

Parallel random public call.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

Definition at line 522 of file random\_shuffle.h.

References `__parallel_random_shuffle_drs()`.

#### 4.6.4.32 `__parallel_random_shuffle_drs()`

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__parallel_random_shuffle_drs (
    _RAIter __begin,
    _RAIter __end,
    typename std::iterator_traits< _RAIter >::difference_type __n,
    _ThreadIndex __num_threads,
    _RandomNumberGenerator & __rng )
```

Main parallel random shuffle step.

##### Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__n</code>	Length of sequence.
<code>__num_threads</code>	Number of threads to use.
<code>__rng</code>	Random number generator to use.

Definition at line 265 of file `random_shuffle.h`.

References `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end`, `__parallel_random_shuffle_drs_pu()`, `__rd_log2()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `_GLIBCXX_CALL`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::__M_bin_proc`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_bins_begin`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::__M_dist`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::__M_num_bins`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::__M_num_bits`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_num_threads`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_sd`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_seed`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::__M_starts`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::__M_temporaries`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::Settings::TLB_size`.

Referenced by `__parallel_random_shuffle()`.

#### 4.6.4.33 `__parallel_random_shuffle_drs_pu()`

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__parallel_random_shuffle_drs_pu (
    _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus )
```

Random shuffle code executed by each thread.

## Parameters

<code>__pus</code>	Array of thread-local data records.
--------------------	-------------------------------------

Definition at line 122 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bits`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_num_threads`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_sd`, `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_seed`, `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_starts`, and `std::partial_sum()`.

Referenced by `__parallel_random_shuffle_drs()`.

## 4.6.4.34 \_\_parallel\_sort() [1/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    multiway_mergesort_tag __parallelism) [inline]
```

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

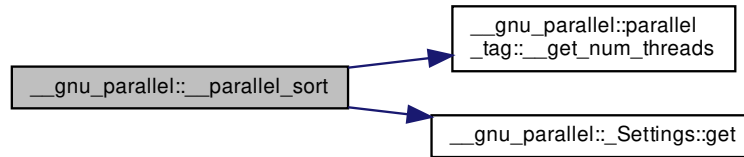
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 75 of file `sort.h`.

References `__gnu_parallel::parallel_tag::_get_num_threads()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



#### 4.6.4.35 `__parallel_sort()` [2/7]

```

template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    multiway_mergesort_exact_tag __parallelism ) [inline]
  
```

Choose multiway mergesort with exact splitting, for parallel sorting.

##### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

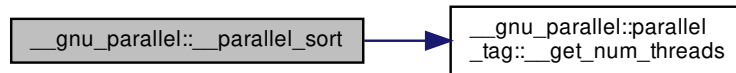
##### Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 99 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



#### 4.6.4.36 \_\_parallel\_sort() [3/7]

```

template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    multiway_mergesort_sampling_tag __parallelism ) [inline]
  
```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

##### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

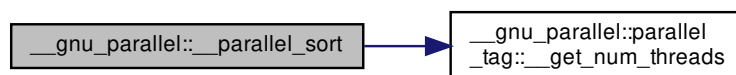
##### Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 120 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:





**4.6.4.37** `__parallel_sort()` [4/7]

```
template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    quicksort_tag __parallelism ) [inline]
```

Choose quicksort for parallel sorting.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

**Template Parameters**

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 140 of file sort.h.

References `_GLIBCXX_CALL`.

**4.6.4.38** `__parallel_sort()` [5/7]

```
template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    balanced_quicksort_tag __parallelism ) [inline]
```

Choose balanced quicksort for parallel sorting.

**Parameters**

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 161 of file sort.h.

References `_GLIBCXX_CALL`.

## 4.6.4.39 \_\_parallel\_sort() [6/7]

```
template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    default_parallel_tag __parallelism ) [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

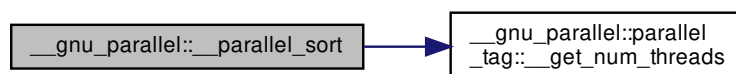
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 183 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



4.6.4.40 `__parallel_sort()` [7/7]

```
template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    parallel_tag __parallelism ) [inline]
```

Choose a parallel sorting algorithm.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

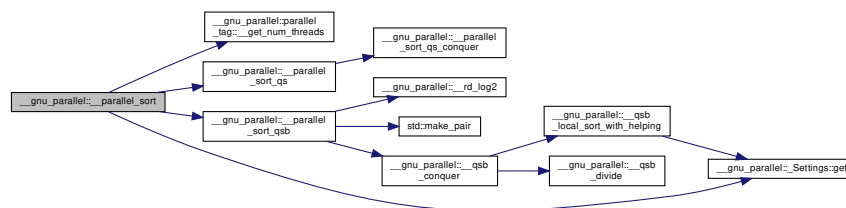
## Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 203 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::Settings::get()`.

Here is the call graph for this function:

4.6.4.41 `__parallel_sort_qs()`

```
template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort_qs (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    ThreadIndex __num_threads )
```

Unbalanced quicksort main call.

## Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator input sequence, ignored.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 154 of file quicksort.h.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

4.6.4.42 `__parallel_sort_qs_conquer()`

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort_qs_conquer (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads )
```

Unbalanced quicksort conquer step.

## Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 101 of file quicksort.h.

Referenced by `__parallel_sort_qs()`.

4.6.4.43 `__parallel_sort_qs_divide()`

```
template<typename _RAIter , typename _Compare >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_sort_qs_divide (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    typename std::iterator_traits<_RAIter >::difference_type __pivot_rank,
    typename std::iterator_traits<_RAIter >::difference_type __num_samples,
    _ThreadIndex __num_threads )
```

Unbalanced quicksort divide step.

## Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__pivot_rank</code>	Desired <code>__rank</code> of the pivot.
<code>__num_samples</code>	Choose pivot from that many samples.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References `std::min()`.

4.6.4.44 `__parallel_sort_qsb()`

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort_qsb (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads )
```

Top-level quicksort routine.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 433 of file balanced\_quicksort.h.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, `__gnu_parallel::QSBThreadLocal<_RAIter>::__M_↔elements_leftover`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

4.6.4.45 `__parallel_unique_copy()` [1/2]

```
template<typename _IIter , class _OutputIterator , class _BinaryPredicate >
_OutputIterator __gnu_parallel::__parallel_unique_copy (
    _IIter __first,
    _IIter __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred )
```

Parallel `std::unique_copy()`, w/\_o explicit equality predicate.

**Parameters**

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result __sequence.
<code>__binary_pred</code>	Equality predicate.

**Returns**

End iterator of result \_\_sequence.

Definition at line 50 of file `unique_copy.h`.

References `_GLIBCXX_CALL`.

Referenced by `__parallel_unique_copy()`.

**4.6.4.46 \_\_parallel\_unique\_copy()** [2/2]

```
template<typename _IIter , class _OutputIterator >
_OutputIterator __gnu_parallel::__parallel_unique_copy (
    _IIter __first,
    _IIter __last,
    _OutputIterator __result ) [inline]
```

Parallel `std::unique_copy()`, without explicit equality predicate.

**Parameters**

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result __sequence.

**Returns**

End iterator of result \_\_sequence.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

#### 4.6.4.47 \_\_qsb\_conquer()

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__qsb_conquer (
    _QSBThreadLocal< _RAIter > ** __tls,
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __iam,
    _ThreadIndex __num_threads,
    bool __parent_wait )
```

Quicksort conquer step.

##### Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 174 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, and `__gnu_parallel::_QSBThreadLocal< _RAIter >::↵M_initial`.

Referenced by `__parallel_sort_qsb()`.

#### 4.6.4.48 \_\_qsb\_divide()

```
template<typename _RAIter , typename _Compare >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__qsb_divide (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads )
```

Balanced quicksort divide step.

##### Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

## Precondition

```
(__end-__begin)>=1
```

Definition at line 103 of file `balanced_quicksort.h`.

Referenced by `__qsb_conquer()`.

4.6.4.49 `__qsb_local_sort_with_helping()`

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__qsb_local_sort_with_helping (
    __QSBThreadLocal< _RAIter > ** __tls,
    _Compare & __comp,
    __ThreadIndex __iam,
    bool __wait )
```

Quicksort step doing load-balanced local sort.

## Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.

Definition at line 250 of file `balanced_quicksort.h`.

References `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_initial`, `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_num_threads`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::sort_qsb_base_case_maximal←_n`.

Referenced by `__qsb_conquer()`.

4.6.4.50 `__random_number_pow2()`

```
template<typename _RandomNumberGenerator >
int __gnu_parallel::__random_number_pow2 (
    int __logp,
    _RandomNumberGenerator & __rng ) [inline]
```

Generate a random number in  $[0, 2^{\text{__logp}})$ .

## Parameters

<code>__logp</code>	Logarithm (basis 2) of the upper range <code>__bound</code> .
<code>__rng</code>	Random number generator to use.



Definition at line 115 of file random\_shuffle.h.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

#### 4.6.4.51 `__rd_log2()`

```
template<typename _Size >
_Size __gnu_parallel::__rd_log2 (
    _Size __n ) [inline]
```

Calculates the rounded-down logarithm of `__n` for base 2.

##### Parameters

<code>↔</code> <code>__n</code>	Argument.
------------------------------------	-----------

##### Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file parallel/base.h.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_↔random_shuffle()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `multiseq_selection()`.

#### 4.6.4.52 `__round_up_to_pow2()`

```
template<typename _Tp >
_Tp __gnu_parallel::__round_up_to_pow2 (
    _Tp __x )
```

Round up to the next greater power of 2.

##### Parameters

<code>↔</code> <code>__x</code>	<code>__Integer to round up</code>
------------------------------------	------------------------------------

Definition at line 248 of file random\_shuffle.h.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

## 4.6.4.53 \_\_search\_template()

```
template<typename __RAIter1 , typename __RAIter2 , typename _Pred >
__RAIter1 __gnu_parallel::__search_template (
    __RAIter1 __begin1,
    __RAIter1 __end1,
    __RAIter2 __begin2,
    __RAIter2 __end2,
    _Pred __pred )
```

Parallel std::search.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__pred</code>	Find predicate.

## Returns

Place of finding in first sequences.

Definition at line 81 of file search.h.

References `__calc_borders()`, `__equally_split()`, `_GLIBCXX_CALL`, and `std::min()`.

## 4.6.4.54 \_\_sequential\_multiway\_merge()

```
template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename
_DifferenceTp , typename _Compare >
__RAIter3 __gnu_parallel::__sequential_multiway_merge (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
    _DifferenceTp __length,
    _Compare __comp )
```

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.
<code>__sentinel</code>	The sequences have a sentinel element.

**Returns**

End iterator of output sequence.

Definition at line 920 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

**4.6.4.55 `__sequential_random_shuffle()`**

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__sequential_random_shuffle (
    _RAIter __begin,
    _RAIter __end,
    _RandomNumberGenerator & __rng )
```

Sequential cache-efficient random shuffle.

**Parameters**

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

Definition at line 410 of file random\_shuffle.h.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, `std::partial_sum()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.56 `__shrink()`**

```
template<typename _IIter >
void __gnu_parallel::__shrink (
    std::vector<_IIter > & __os_starts,
    size_t & __count_to_two,
    size_t & __range_length )
```

Combines two ranges into one and thus halves the number of ranges.

**Parameters**

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.

Definition at line 70 of file list\_partition.h.

References `std::vector<_Tp, _Alloc>::size()`.

Referenced by `__shrink_and_double()`.

#### 4.6.4.57 \_\_shrink\_and\_double()

```
template<typename _IIter >
void __gnu_parallel::__shrink_and_double (
    std::vector<_IIter > & __os_starts,
    size_t & __count_to_two,
    size_t & __range_length,
    const bool __make_twice )
```

Shrinks and doubles the ranges.

##### Parameters

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.
<code>__make_twice</code>	Whether the <code>__os_starts</code> is allowed to be grown or not

Definition at line 50 of file list\_partition.h.

References `__shrink()`, `std::vector<_Tp, _Alloc>::resize()`, and `std::vector<_Tp, _Alloc>::size()`.

Referenced by `list_partition()`.

#### 4.6.4.58 \_\_yield()

```
void __gnu_parallel::__yield ( ) [inline]
```

Yield control to another thread, without waiting for the end of the time slice.

Definition at line 121 of file parallel/compatibility.h.

#### 4.6.4.59 list\_partition()

```
template<typename _IIter , typename _FunctorType >
size_t __gnu_parallel::list_partition (
    const _IIter __begin,
    const _IIter __end,
    _IIter * __starts,
    size_t * __lengths,
    const int __num_parts,
    _FunctorType & __f,
    int __oversampling = 0 )
```

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

##### Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__starts</code>	Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts [ __num_parts ]</code> contains the end iterator of the sequence.
<code>__lengths</code>	Length of the resulting parts.
<code>__num_parts</code>	Number of parts to split the sequence into.
<code>__f</code>	Functor to be applied to each element by traversing <code>__it</code>
<code>__oversampling</code>	Oversampling factor. If 0, then the partitions will differ in at most $\sqrt{\text{end} - \text{begin}}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1/(\text{oversampling} \cdot \text{num\_parts})$

##### Returns

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector< _Tp, _Alloc >::size()`.

#### 4.6.4.60 max()

```
template<typename _Tp >
const _Tp& __gnu_parallel::max (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

## 4.6.4.61 min()

```
template<typename _Tp >
const _Tp& __gnu_parallel::min (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

## 4.6.4.62 multiseq\_partition()

```
template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare >
void __gnu_parallel::multiseq_partition (
    _RanSeqs __begin_seqs,
    _RanSeqs __end_seqs,
    _RankType __rank,
    _RankIterator __begin_offsets,
    _Compare __comp = std::less< typename std::iterator_traits<typename std::iterator_↵
traits<_RanSeqs>::value_type:: first_type>::value_type>() )
```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

## Parameters

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__begin_offsets</code>	A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> .
<code>__comp</code>	The ordering functor, defaults to <code>std::less&lt;_Tp&gt;</code> .

Definition at line 122 of file `multiseq_selection.h`.

References `_GLIBCXX_CALL`, and `std::distance()`.

## 4.6.4.63 multiseq\_selection()

```
template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare >
_Tp __gnu_parallel::multiseq_selection (
    _RanSeqs __begin_seqs,
    _RanSeqs __end_seqs,
```

```

_RankType __rank,
_RankType & __offset,
_Compare __comp = std::less<_Tp>() )

```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

#### Parameters

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__offset</code>	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
<code>__comp</code>	The ordering functor, defaults to <code>std::less</code> .

Definition at line 388 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `std::__sample()`, `_GLIBCXX_CALL`, `std::distance()`, `std::make_pair()`, and `std::max()`.

#### 4.6.4.64 multiway\_merge()

```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename ↵
_Compare >
_RAIterOut __gnu_parallel::multiway_merge (
    _RAIterPairIterator __seqs_begin,
    _RAIterPairIterator __seqs_end,
    _RAIterOut __target,
    _DifferenceTp __length,
    _Compare __comp,
    __gnu_parallel::sequential_tag )

```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable

- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 10; ++__j)
        sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                     sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

See also

`stable_multiway_merge`

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
return `__value` - `__target` = min(`__length`, number of elements in all sequences).

Template Parameters

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

Parameters

<code>__seqs_begin</code>	<code>__begin</code> of sequence <code>__sequence</code>
<code>__seqs_end</code>	<code>__M_end</code> of sequence <code>__sequence</code>



## Parameters

<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

## Returns

`_M_end` iterator of output sequence

Definition at line 1418 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

4.6.4.65 `multiway_merge_3_variant()`

```
template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename
_RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_3_variant (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _DifferenceTp __length,
    _Compare __comp )
```

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

## Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 241 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.66 multiway\_merge\_4\_variant()**

```
template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename
_RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_4_variant (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _DifferenceTp __length,
    _Compare __comp )
```

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

**Parameters**

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 360 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

#### 4.6.4.67 multiway\_merge\_exact\_splitting()

```
template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
void __gnu_parallel::multiway_merge_exact_splitting (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _DifferenceType __length,
    _DifferenceType __total_length,
    _Compare __comp,
    std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces )
```

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1120 of file multiway\_merge.h.

Referenced by \_\_parallel\_merge\_advance().

#### 4.6.4.68 multiway\_merge\_loser\_tree()

```
template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp ,
typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _DifferenceTp __length,
    _Compare __comp )
```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a LoserTree class as selected by \_LT.

Stability is selected through the used LoserTree class \_LT.

At least one non-empty sequence is required.

##### Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 491 of file multiway\_merge.h.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

**4.6.4.69 multiway\_merge\_loser\_tree\_sentinel()**

```
template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _↵
DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
    _DifferenceTp __length,
    _Compare __comp )
```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

**Template Parameters**

<i>UnguardedLoserTree</i>	<code>_Loser Tree</code> variant to use for the unguarded merging.
---------------------------	--

**Parameters**

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 662 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

#### 4.6.4.70 multiway\_merge\_loser\_tree\_unguarded()

```
template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp ,
typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
    _DifferenceTp __length,
    _Compare __comp )
```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the LoserTree class `_LT`.

Stability is selected by the used LoserTrees.

##### Precondition

No input will run out of elements during the merge.

##### Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

##### Returns

End iterator of output sequence.

Definition at line 574 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

#### 4.6.4.71 multiway\_merge\_sampling\_splitting()

```
template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
void __gnu_parallel::multiway_merge_sampling_splitting (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _DifferenceType __length,
    _DifferenceType __total_length,
```

```

    _Compare __comp,
    std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces )

```

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1035 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

#### 4.6.4.72 `multiway_merge_sentinels()`

```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _
_Compare >
_RAIterOut __gnu_parallel::multiway_merge_sentinels (
    _RAIterPairIterator __seqs_begin,
    _RAIterPairIterator __seqs_end,
    _RAIterOut __target,
    _DifferenceTp __length,
    _Compare __comp,
    __gnu_parallel::sequential_tag )

```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```

int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 11; ++__j)
        sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                    sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);

```

### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

### Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
 return `__value` - `__target` = min(`__length`, number of elements in all sequences).

### See also

`stable_multiway_merge_sentinels`

### Template Parameters

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

### Parameters

<code>__seqs_begin</code>	<code>__begin</code> of sequence <code>__sequence</code>
<code>__seqs_end</code>	<code>_M_end</code> of sequence <code>__sequence</code>
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

**Returns**

\_M\_end iterator of output sequence

Definition at line 1782 of file multiway\_merge.h.

References \_\_sequential\_multiway\_merge(), and \_GLIBCXX\_CALL.

**4.6.4.73 parallel\_multiway\_merge()**

```
template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename
_DifferenceTp , typename _Splitter , typename _Compare >
_RAIter3 __gnu_parallel::parallel_multiway_merge (
    _RAIterIterator __seqs_begin,
    _RAIterIterator __seqs_end,
    _RAIter3 __target,
    _Splitter __splitter,
    _DifferenceTp __length,
    _Compare __comp,
    _ThreadIndex __num_threads )
```

Parallel multi-way merge routine.

The \_GLIBCXX\_PARALLEL\_DECISION is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

**Template Parameters**

<i>_Splitter</i>	functor to split input (either __exact or sampling based)
<i>__stable</i>	Stable merging incurs a performance penalty.
<i>__sentinel</i>	Ignored.

**Parameters**

<i>__seqs_begin</i>	Begin iterator of iterator pair input sequence.
<i>__seqs_end</i>	End iterator of iterator pair input sequence.
<i>__target</i>	Begin iterator of output sequence.
<i>__comp</i>	Comparator.
<i>__length</i>	Maximum length to merge, possibly larger than the number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 1225 of file multiway\_merge.h.

Referenced by \_\_parallel\_merge\_advance().



#### 4.6.4.74 parallel\_sort\_mwms()

```
template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
void __gnu_parallel::parallel_sort_mwms (
    _RAIter __begin,
    _RAIter __end,
    _Compare __comp,
    _ThreadIndex __num_threads )
```

PMWMS main call.

##### Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads to use.

Definition at line 395 of file multiway\_mergesort.h.

References `_GLIBCXX_CALL`.

#### 4.6.4.75 parallel\_sort\_mwms\_pu()

```
template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
void __gnu_parallel::parallel_sort_mwms_pu (
    _PMWMSSortingData< _RAIter > * __sd,
    _Compare & __comp )
```

PMWMS code executed by each thread.

##### Parameters

<code>__sd</code>	Pointer to algorithm data.
<code>__comp</code>	Comparator.

Definition at line 308 of file multiway\_mergesort.h.

References `__gnu_parallel::_PMWMSSortingData<_RAIter>::_M_num_threads`, `__gnu_parallel::_PMWMSSortingData<_RAIter>::_M_source`, `__gnu_parallel::_PMWMSSortingData<_RAIter>::_M_starts`, `__gnu_parallel::_PMWMSSortingData<_RAIter>::_M_temporary`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, `__gnu_parallel::_Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

#### 4.6.5 Variable Documentation

## 4.6.5.1 \_CASable\_bits

```
const int __gnu_parallel::_CASable_bits [static]
```

Number of bits of \_CASable.

Definition at line 130 of file types.h.

Referenced by \_\_decode2(), and \_\_encode2().

## 4.6.5.2 \_CASable\_mask

```
const _CASable __gnu_parallel::_CASable_mask [static]
```

\_CASable with the right half of bits set to 1.

Definition at line 133 of file types.h.

Referenced by \_\_decode2().

## 4.7 \_\_gnu\_pbds Namespace Reference

## Classes

- struct [associative\\_tag](#)
- class [basic\\_branch](#)
- struct [basic\\_branch\\_tag](#)
- class [basic\\_hash\\_table](#)
- struct [basic\\_hash\\_tag](#)
- struct [basic\\_invalidation\\_guarantee](#)
- struct [binary\\_heap\\_tag](#)
- struct [binomial\\_heap\\_tag](#)
- class [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)
- class [cc\\_hash\\_table](#)
- struct [cc\\_hash\\_tag](#)
- struct [container\\_error](#)
- struct [container\\_tag](#)
- struct [container\\_traits](#)
- struct [container\\_traits\\_base](#)
- struct [container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [container\\_traits\\_base< rb\\_tree\\_tag >](#)

- struct [container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< thin\\_heap\\_tag >](#)
- class [direct\\_mask\\_range\\_hashing](#)
- class [direct\\_mod\\_range\\_hashing](#)
- class [gp\\_hash\\_table](#)
- struct [gp\\_hash\\_tag](#)
- class [hash\\_exponential\\_size\\_policy](#)
- class [hash\\_load\\_check\\_resize\\_trigger](#)
- class [hash\\_prime\\_size\\_policy](#)
- class [hash\\_standard\\_resize\\_policy](#)
- struct [insert\\_error](#)
- struct [join\\_error](#)
- class [linear\\_probe\\_fn](#)
- class [list\\_update](#)
- struct [list\\_update\\_tag](#)
- class [lu\\_counter\\_policy](#)
- class [lu\\_move\\_to\\_front\\_policy](#)
- struct [null\\_node\\_update](#)
- struct [null\\_type](#)
- struct [ov\\_tree\\_tag](#)
- struct [pairing\\_heap\\_tag](#)
- struct [pat\\_trie\\_tag](#)
- struct [point\\_invalidation\\_guarantee](#)
- class [priority\\_queue](#)
- struct [priority\\_queue\\_tag](#)
- class [quadratic\\_probe\\_fn](#)
- struct [range\\_invalidation\\_guarantee](#)
- struct [rb\\_tree\\_tag](#)
- struct [rc\\_binomial\\_heap\\_tag](#)
- struct [resize\\_error](#)
- class [sample\\_probe\\_fn](#)
- class [sample\\_range\\_hashing](#)
- class [sample\\_ranged\\_hash\\_fn](#)
- class [sample\\_ranged\\_probe\\_fn](#)
- class [sample\\_resize\\_policy](#)
- class [sample\\_resize\\_trigger](#)
- class [sample\\_size\\_policy](#)
- class [sample\\_tree\\_node\\_update](#)
- struct [sample\\_trie\\_access\\_traits](#)
- class [sample\\_trie\\_node\\_update](#)
- struct [sample\\_update\\_policy](#)
- struct [sequence\\_tag](#)
- struct [splay\\_tree\\_tag](#)
- struct [string\\_tag](#)
- struct [thin\\_heap\\_tag](#)
- class [tree](#)
- class [tree\\_order\\_statistics\\_node\\_update](#)
- struct [tree\\_tag](#)
- class [trie](#)
- class [trie\\_order\\_statistics\\_node\\_update](#)

- class [trie\\_prefix\\_search\\_node\\_update](#)
- struct [trie\\_string\\_access\\_traits](#)
- struct [trie\\_tag](#)
- struct [trivial\\_iterator\\_tag](#)

#### Typedefs

- typedef void [trivial\\_iterator\\_difference\\_type](#)

#### Functions

- void [\\_\\_throw\\_container\\_error](#) ()
- void [\\_\\_throw\\_insert\\_error](#) ()
- void [\\_\\_throw\\_join\\_error](#) ()
- void [\\_\\_throw\\_resize\\_error](#) ()

#### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 `__gnu_profile` Namespace Reference

#### Classes

- class [\\_\\_container\\_size\\_info](#)
- class [\\_\\_container\\_size\\_stack\\_info](#)
- class [\\_\\_hashfunc\\_info](#)
- class [\\_\\_hashfunc\\_stack\\_info](#)
- class [\\_\\_list2vector\\_info](#)
- class [\\_\\_map2umap\\_info](#)
- class [\\_\\_map2umap\\_stack\\_info](#)
- class [\\_\\_object\\_info\\_base](#)
- struct [\\_\\_reentrance\\_guard](#)
- class [\\_\\_stack\\_hash](#)
- class [\\_\\_trace\\_base](#)
- class [\\_\\_trace\\_container\\_size](#)
- class [\\_\\_trace\\_hash\\_func](#)
- class [\\_\\_trace\\_hashtable\\_size](#)
- class [\\_\\_trace\\_map2umap](#)
- class [\\_\\_trace\\_vector\\_size](#)
- class [\\_\\_trace\\_vector\\_to\\_list](#)
- class [\\_\\_vector2list\\_info](#)
- class [\\_\\_vector2list\\_stack\\_info](#)
- struct [\\_\\_warning\\_data](#)

## Typedefs

- typedef std::vector< \_\_cost\_factor \* > **\_\_cost\_factor\_vector**
- typedef std::unordered\_map< [std::string](#), [std::string](#) > **\_\_env\_t**
- typedef void \* **\_\_instruction\_address\_t**
- typedef std::vector< \_\_instruction\_address\_t > **\_\_stack\_npt**
- typedef \_\_stack\_npt \* **\_\_stack\_t**
- typedef std::vector< [\\_\\_warning\\_data](#) > **\_\_warning\_vector\_t**

## Enumerations

- enum **\_\_state\_type** { **\_\_ON**, **\_\_OFF**, **\_\_INVALID** }

## Functions

- std::size\_t **\_\_env\_to\_size\_t** (const char \* \_\_env\_var, std::size\_t \_\_default\_value)
- template<typename \_InputIterator, typename \_Function >  
\_Function **\_\_for\_each** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Function \_\_f)
- **\_\_stack\_t** **\_\_get\_stack** ()
- template<typename \_Container >  
void **\_\_insert\_top\_n** (\_Container &\_\_output, const typename \_Container::value\_type &\_\_value, typename \_Container::size\_type \_\_n)
- bool **\_\_is\_invalid** ()
- bool **\_\_is\_off** ()
- bool **\_\_is\_on** ()
- int **\_\_log2** (std::size\_t \_\_size)
- int **\_\_log\_magnitude** (float \_\_f)
- float **\_\_map\_erase\_cost** (std::size\_t \_\_size)
- float **\_\_map\_find\_cost** (std::size\_t \_\_size)
- float **\_\_map\_insert\_cost** (std::size\_t \_\_size)
- std::size\_t **\_\_max\_mem** ()
- FILE \* **\_\_open\_output\_file** (const char \* \_\_extension)
- bool [\\_\\_profcxx\\_init](#) ()
- void **\_\_profcxx\_init\_unconditional** ()
- void **\_\_read\_cost\_factors** ()
- template<typename \_ForwardIterator, typename \_Tp >  
\_ForwardIterator **\_\_remove** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_value)
- void [\\_\\_report](#) ()
- void **\_\_report\_and\_free** ()
- void **\_\_set\_cost\_factors** ()
- void **\_\_set\_max\_mem** ()
- void **\_\_set\_max\_stack\_trace\_depth** ()
- void **\_\_set\_max\_warn\_count** ()
- void **\_\_set\_trace\_path** ()
- std::size\_t **\_\_size** (\_\_stack\_t \_\_stack)
- std::size\_t **\_\_stack\_max\_depth** ()
- template<typename \_Container >  
void **\_\_top\_n** (const \_Container &\_\_input, \_Container &\_\_output, typename \_Container::size\_type \_\_n)
- [\\_\\_hashfunc\\_info](#) \* **\_\_trace\_hash\_func\_construct** ()

- `void __trace_hash_func_destruct ( __hashfunc_info *, std::size_t, std::size_t, std::size_t)`
- `void __trace_hash_func_free ()`
- `void __trace_hash_func_init ()`
- `void __trace_hash_func_report (FILE * __f, __warning_vector_t & __warnings)`
- `__container_size_info * __trace_hashtable_size_construct (std::size_t)`
- `void __trace_hashtable_size_destruct ( __container_size_info *, std::size_t, std::size_t)`
- `void __trace_hashtable_size_free ()`
- `void __trace_hashtable_size_init ()`
- `void __trace_hashtable_size_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_hashtable_size_resize ( __container_size_info *, std::size_t, std::size_t)`
- `__list2slist_info * __trace_list_to_slist_construct ()`
- `void __trace_list_to_slist_destruct ( __list2slist_info *)`
- `void __trace_list_to_slist_free ()`
- `void __trace_list_to_slist_init ()`
- `void __trace_list_to_slist_operation ( __list2slist_info *)`
- `void __trace_list_to_slist_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_list_to_slist_rewind ( __list2slist_info *)`
- `__list2vector_info * __trace_list_to_vector_construct ()`
- `void __trace_list_to_vector_destruct ( __list2vector_info *)`
- `void __trace_list_to_vector_free ()`
- `void __trace_list_to_vector_init ()`
- `void __trace_list_to_vector_insert ( __list2vector_info *, std::size_t, std::size_t)`
- `void __trace_list_to_vector_invalid_operator ( __list2vector_info *)`
- `void __trace_list_to_vector_iterate ( __list2vector_info *, int)`
- `void __trace_list_to_vector_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __trace_list_to_vector_resize ( __list2vector_info *, std::size_t, std::size_t)`
- `__map2umap_info * __trace_map_to_unordered_map_construct ()`
- `void __trace_map_to_unordered_map_destruct ( __map2umap_info *)`
- `void __trace_map_to_unordered_map_erase ( __map2umap_info *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_find ( __map2umap_info *, std::size_t)`
- `void __trace_map_to_unordered_map_free ()`
- `void __trace_map_to_unordered_map_init ()`
- `void __trace_map_to_unordered_map_insert ( __map2umap_info *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_invalidate ( __map2umap_info *)`
- `void __trace_map_to_unordered_map_iterate ( __map2umap_info *, std::size_t)`
- `void __trace_map_to_unordered_map_iterate ( __map2umap_info * __info, int)`
- `void __trace_map_to_unordered_map_report (FILE * __f, __warning_vector_t & __warnings)`
- `template<typename __object_info, typename __stack_info >`  
`void __trace_report ( __trace_base< __object_info, __stack_info > * __cont, FILE * __f, __warning_vector_t & __warnings)`
- `__container_size_info * __trace_vector_size_construct (std::size_t)`
- `void __trace_vector_size_destruct ( __container_size_info *, std::size_t, std::size_t)`
- `void __trace_vector_size_free ()`
- `void __trace_vector_size_init ()`
- `void __trace_vector_size_report (FILE *, __warning_vector_t &)`
- `void __trace_vector_size_resize ( __container_size_info *, std::size_t, std::size_t)`
- `__vector2list_info * __trace_vector_to_list_construct ()`
- `void __trace_vector_to_list_destruct ( __vector2list_info *)`
- `void __trace_vector_to_list_free ()`
- `void __trace_vector_to_list_init ()`
- `void __trace_vector_to_list_insert ( __vector2list_info *, std::size_t, std::size_t)`

- void `__trace_vector_to_list_invalid_operator` (`__vector2list_info *`)
- void `__trace_vector_to_list_iterate` (`__vector2list_info *`, int)
- void `__trace_vector_to_list_report` (FILE \*, `__warning_vector_t &`)
- void `__trace_vector_to_list_resize` (`__vector2list_info *`, `std::size_t`, `std::size_t`)
- bool `__turn` (`__state_type __s`)
- bool `__turn_off` ()
- bool `__turn_on` ()
- void `__write` (FILE \* `__f`, `__stack_t __stack`)
- void `__write_cost_factors` ()
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__state_type`, `__state`, `__INVALID`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_hash_func *`, `_S_hash_func`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_hashtable_size *`, `_S_hashtable_size`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_map2umap *`, `_S_map2umap`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_vector_size *`, `_S_vector_size`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_vector_to_list *`, `_S_vector_to_list`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_list_to_slist *`, `_S_list_to_slist`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__trace_list_to_vector *`, `_S_list_to_vector`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_shift_cost_factor`, {"\_\_vector\_shift\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_iterate_cost_factor`, {"\_\_vector\_iterate\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__vector_resize_cost_factor`, {"\_\_vector\_resize\_cost\_factor", 1.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_shift_cost_factor`, {"\_\_list\_shift\_cost\_factor", 0.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_iterate_cost_factor`, {"\_\_list\_iterate\_cost\_factor", 10.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_resize_cost_factor`, {"\_\_list\_resize\_cost\_factor", 0.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_insert_cost_factor`, {"\_\_map\_insert\_cost\_factor", 1.5})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_erase_cost_factor`, {"\_\_map\_erase\_cost\_factor", 1.5})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_find_cost_factor`, {"\_\_map\_find\_cost\_factor", 1})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_iterate_cost_factor`, {"\_\_map\_iterate\_cost\_factor", 2.3})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_insert_cost_factor`, {"\_\_umap\_insert\_cost\_factor", 12.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_erase_cost_factor`, {"\_\_umap\_erase\_cost\_factor", 12.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_find_cost_factor`, {"\_\_umap\_find\_cost\_factor", 10.0})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_iterate_cost_factor`, {"\_\_umap\_iterate\_cost\_factor", 1.7})
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor_vector *`, `__cost_factors`, 0)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (const char \*, `_S_trace_file_name`, `_GLIBCXX_PROFILE_TRACE_PATH_ROOT`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_warn_count`, `_GLIBCXX_PROFILE_MAX_WARN_COUNT`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_stack_depth`, `_GLIBCXX_PROFILE_MAX_STACK_DEPTH`)
- `_GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_mem`, `_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__env_t`, `__env`)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__gnu_cxx::__mutex`, `__global_mutex`)

#### 4.8.1 Detailed Description

GNU profile code for public use.

#### 4.8.2 Typedef Documentation

##### 4.8.2.1 `__env_t`

```
typedef std:: unordered_map<std::string, std::string> __gnu_profile::__env_t
```

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is `libstdcxx-profile.conf`. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 65 of file `profiler_trace.h`.

#### 4.8.3 Function Documentation

##### 4.8.3.1 `__profcxx_init()`

```
bool __gnu_profile::__profcxx_init ( ) [inline]
```

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 653 of file `profiler_trace.h`.

##### 4.8.3.2 `__report()`

```
void __gnu_profile::__report ( ) [inline]
```

Final report method, registered with **atexit**.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in `profiler.h`. However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 448 of file `profiler_trace.h`.



#### 4.8.3.3 `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA()`

```
__gnu_profile::_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA (
    __gnu_cxx::__mutex ,
    __global_mutex )
```

Master lock.

### 4.9 `__gnu_sequential` Namespace Reference

#### 4.9.1 Detailed Description

GNU sequential classes for public use.

### 4.10 `abi` Namespace Reference

#### 4.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

### 4.11 `std` Namespace Reference

#### Namespaces

- [\\_\\_debug](#)
- [\\_\\_detail](#)
- [\\_\\_parallel](#)
- [\\_\\_profile](#)
- [chrono](#)
- [decimal](#)
- [placeholders](#)
- [regex\\_constants](#)
- [rel\\_ops](#)
- [this\\_thread](#)
- [tr1](#)
- [tr2](#)

## Classes

- struct [\\_\\_add\\_pointer\\_helper](#)
- struct [\\_\\_allocated\\_ptr](#)
- struct [\\_\\_atomic\\_base](#)
- struct [\\_\\_atomic\\_base< \\_PTp \\* >](#)
- struct [\\_\\_atomic\\_flag\\_base](#)
- class [\\_\\_basic\\_future](#)
- class [\\_\\_codecvt\\_abstract\\_base](#)
- class [\\_\\_ctype\\_abstract\\_base](#)
- struct [\\_\\_detector](#)
- struct [\\_\\_detector< \\_Default, \\_\\_void\\_t< \\_Op< \\_Args... > >, \\_Op, \\_Args... >](#)
- struct [\\_\\_future\\_base](#)
- struct [\\_\\_is\\_location\\_invariant](#)
- struct [\\_\\_is\\_nullptr\\_t](#)
- struct [\\_\\_is\\_trivially\\_copy\\_assignable\\_impl](#)
- struct [\\_\\_is\\_trivially\\_copy\\_constructible\\_impl](#)
- struct [\\_\\_is\\_trivially\\_move\\_assignable\\_impl](#)
- struct [\\_\\_is\\_trivially\\_move\\_constructible\\_impl](#)
- struct [\\_\\_is\\_tuple\\_like\\_impl< std::pair< \\_T1, \\_T2 > >](#)
- struct [\\_\\_iterator\\_traits](#)
- struct [\\_\\_numeric\\_limits\\_base](#)
- class [\\_\\_shared\\_mutex\\_cv](#)
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- struct [\\_Bind](#)
- struct [\\_Bind\\_result](#)
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)
- struct [\\_Enable\\_copy\\_move](#)
- struct [\\_Enable\\_default\\_constructor](#)
- struct [\\_Enable\\_destructor](#)
- struct [\\_Enable\\_special\\_members](#)
- class [\\_Function\\_base](#)
- struct [\\_Fwd\\_list\\_base](#)
- struct [\\_Fwd\\_list\\_const\\_iterator](#)
- struct [\\_Fwd\\_list\\_iterator](#)
- struct [\\_Fwd\\_list\\_node](#)
- struct [\\_Fwd\\_list\\_node\\_base](#)
- class [\\_Hashtable](#)
- class [\\_List\\_base](#)
- struct [\\_List\\_const\\_iterator](#)
- struct [\\_List\\_iterator](#)
- struct [\\_List\\_node](#)
- struct [\\_Maybe\\_get\\_result\\_type](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1 >](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1, \\_T2 >](#)
- class [\\_Mu](#)
- class [\\_Mu< \\_Arg, false, false >](#)

- class `_Mu< _Arg, false, true >`
- class `_Mu< _Arg, true, false >`
- class `_Mu< reference_wrapper< _Tp >, false, false >`
- class `_Not_fn`
- struct `_Placeholder`
- struct `_Reference_wrapper_base`
- struct `_Sp_ebo_helper< _Nm, _Tp, false >`
- struct `_Sp_ebo_helper< _Nm, _Tp, true >`
- class `_Temporary_buffer`
- struct `_Tuple_impl`
- struct `_Tuple_impl< _Idx, _Head, _Tail... >`
- struct `_Vector_base`
- struct `_Weak_result_type`
- struct `_Weak_result_type_impl`
- struct `_Weak_result_type_impl< _Res(*)(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL >`
- struct `_Weak_result_type_impl< _Res(*)(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL >`
- struct `_Weak_result_type_impl< _Res(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL >`
- struct `_Weak_result_type_impl< _Res(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL >`
- struct `add_const`
- struct `add_cv`
- struct `add_lvalue_reference`
- struct `add_rvalue_reference`
- struct `add_volatile`
- struct `adopt_lock_t`
- struct `aligned_storage`
- struct `aligned_union`
- struct `alignment_of`
- class `allocator`
- class `allocator< void >`
- struct `allocator_arg_t`
- struct `allocator_traits`
- struct `allocator_traits< allocator< _Tp > >`
- struct `array`
- struct `atomic`
- struct `atomic< _Tp * >`
- struct `atomic< bool >`
- struct `atomic< char >`
- struct `atomic< char16_t >`
- struct `atomic< char32_t >`
- struct `atomic< int >`
- struct `atomic< long >`
- struct `atomic< long long >`
- struct `atomic< short >`
- struct `atomic< signed char >`
- struct `atomic< unsigned char >`
- struct `atomic< unsigned int >`
- struct `atomic< unsigned long >`
- struct `atomic< unsigned long long >`
- struct `atomic< unsigned short >`
- struct `atomic< wchar_t >`
- struct `atomic_flag`

- class `auto_ptr`
- struct `auto_ptr_ref`
- class `back_insert_iterator`
- class `bad_alloc`
- class `bad_cast`
- class `bad_exception`
- class `bad_function_call`
- class `bad_typeid`
- class `bad_weak_ptr`
- class `basic_filebuf`
- class `basic_fstream`
- class `basic_ifstream`
- class `basic_ios`
- class `basic_iostream`
- class `basic_istream`
- class `basic_istreamstream`
- class `basic_ofstream`
- class `basic_ostream`
- class `basic_ostreamstream`
- class `basic_regex`
- class `basic_streambuf`
- class `basic_string`
- class `basic_stringbuf`
- class `basic_stringstream`
- class `bernoulli_distribution`
- struct `bidirectional_iterator_tag`
- struct `binary_function`
- class `binary_negate`
- class `binder1st`
- class `binder2nd`
- class `binomial_distribution`
- class `bitset`
- class `cauchy_distribution`
- struct `char_traits`
- struct `char_traits< __gnu_cxx::character< _Value, _Int, _St > >`
- struct `char_traits< char >`
- struct `char_traits< wchar_t >`
- class `chi_squared_distribution`
- class `codecvt`
- class `codecvt< _InternT, _ExternT, encoding_state >`
- class `codecvt< char, char, mbstate_t >`
- class `codecvt< char16_t, char, mbstate_t >`
- class `codecvt< char32_t, char, mbstate_t >`
- class `codecvt< wchar_t, char, mbstate_t >`
- class `codecvt_base`
- class `codecvt_byname`
- class `collate`
- class `collate_byname`
- struct `common_type`
- struct `complex`
- struct `complex< double >`

- struct `complex< float >`
- struct `complex< long double >`
- class `condition_variable`
- struct `conditional`
- class `const_mem_fun1_ref_t`
- class `const_mem_fun1_t`
- class `const_mem_fun_ref_t`
- class `const_mem_fun_t`
- class `ctype`
- class `ctype< char >`
- class `ctype< wchar_t >`
- struct `ctype_base`
- class `ctype_byname`
- class `ctype_byname< char >`
- class `decay`
- struct `default_delete`
- struct `default_delete< _Tp[]>`
- struct `defer_lock_t`
- class `deque`
- class `discard_block_engine`
- class `discrete_distribution`
- struct `divides`
- struct `divides< void >`
- class `domain_error`
- struct `enable_if`
- class `enable_shared_from_this`
- struct `equal_to`
- struct `equal_to< void >`
- struct `error_code`
- struct `error_condition`
- class `exception`
- class `exponential_distribution`
- struct `extent`
- class `extreme_value_distribution`
- class `fisher_f_distribution`
- struct `forward_iterator_tag`
- class `forward_list`
- class `fpos`
- class `front_insert_iterator`
- class `function< _Res(_ArgTypes...)>`
- class `future`
- class `future< _Res & >`
- class `future< void >`
- class `future_error`
- class `gamma_distribution`
- class `geometric_distribution`
- struct `greater`
- struct `greater< void >`
- struct `greater_equal`
- struct `greater_equal< void >`
- class `gslice`

- class `gslice_array`
- struct `has_virtual_destructor`
- struct `hash`
- struct `hash< __debug::bitset< _Nb > >`
- struct `hash< __debug::vector< bool, _Alloc > >`
- struct `hash< __gnu_cxx::__u16vstring >`
- struct `hash< __gnu_cxx::__u32vstring >`
- struct `hash< __gnu_cxx::__vstring >`
- struct `hash< __gnu_cxx::__wvstring >`
- struct `hash< __gnu_cxx::throw_value_limit >`
- struct `hash< __gnu_cxx::throw_value_random >`
- struct `hash< __profile::bitset< _Nb > >`
- struct `hash< __profile::vector< bool, _Alloc > >`
- struct `hash< __shared_ptr< _Tp, _Lp > >`
- struct `hash< _Tp * >`
- struct `hash< bool >`
- struct `hash< char >`
- struct `hash< char16_t >`
- struct `hash< char32_t >`
- struct `hash< double >`
- struct `hash< error_code >`
- struct `hash< experimental::shared_ptr< _Tp > >`
- struct `hash< float >`
- struct `hash< int >`
- struct `hash< long >`
- struct `hash< long double >`
- struct `hash< long long >`
- struct `hash< shared_ptr< _Tp > >`
- struct `hash< short >`
- struct `hash< signed char >`
- struct `hash< string >`
- struct `hash< thread::id >`
- struct `hash< type_index >`
- struct `hash< u16string >`
- struct `hash< u32string >`
- struct `hash< unique_ptr< _Tp, _Dp > >`
- struct `hash< unsigned char >`
- struct `hash< unsigned int >`
- struct `hash< unsigned long >`
- struct `hash< unsigned long long >`
- struct `hash< unsigned short >`
- struct `hash< wchar_t >`
- struct `hash< wstring >`
- struct `hash<::bitset< _Nb > >`
- struct `hash<::vector< bool, _Alloc > >`
- class `independent_bits_engine`
- class `indirect_array`
- class `initializer_list`
- struct `input_iterator_tag`
- class `insert_iterator`
- struct `integer_sequence`

- struct [integral\\_constant](#)
- class [invalid\\_argument](#)
- class [ios\\_base](#)
- struct [is\\_abstract](#)
- struct [is\\_arithmetic](#)
- struct [is\\_array](#)
- struct [is\\_assignable](#)
- struct [is\\_base\\_of](#)
- struct [is\\_bind\\_expression](#)
- struct [is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< const \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< const \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< const volatile \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< const volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< volatile \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_class](#)
- struct [is\\_compound](#)
- struct [is\\_const](#)
- struct [is\\_constructible](#)
- struct [is\\_convertible](#)
- struct [is\\_copy\\_assignable](#)
- struct [is\\_copy\\_constructible](#)
- struct [is\\_default\\_constructible](#)
- struct [is\\_destructible](#)
- struct [is\\_empty](#)
- struct [is\\_enum](#)
- struct [is\\_error\\_code\\_enum](#)
- struct [is\\_error\\_code\\_enum< future\\_errc >](#)
- struct [is\\_error\\_condition\\_enum](#)
- struct [is\\_final](#)
- struct [is\\_floating\\_point](#)
- struct [is\\_function](#)
- struct [is\\_fundamental](#)
- struct [is\\_integral](#)
- struct [is\\_literal\\_type](#)
- struct [is\\_lvalue\\_reference](#)
- struct [is\\_member\\_function\\_pointer](#)
- struct [is\\_member\\_object\\_pointer](#)
- struct [is\\_member\\_pointer](#)
- struct [is\\_move\\_assignable](#)
- struct [is\\_move\\_constructible](#)
- struct [is\\_nothrow\\_assignable](#)
- struct [is\\_nothrow\\_constructible](#)
- struct [is\\_nothrow\\_copy\\_assignable](#)
- struct [is\\_nothrow\\_copy\\_constructible](#)
- struct [is\\_nothrow\\_default\\_constructible](#)
- struct [is\\_nothrow\\_destructible](#)
- struct [is\\_nothrow\\_move\\_assignable](#)
- struct [is\\_nothrow\\_move\\_constructible](#)

- struct [is\\_nothrow\\_swappable](#)
- struct [is\\_nothrow\\_swappable\\_with](#)
- struct [is\\_null\\_pointer](#)
- struct [is\\_object](#)
- struct [is\\_placeholder](#)
- struct [is\\_placeholder< \\_Placeholder< \\_Num > >](#)
- struct [is\\_pod](#)
- struct [is\\_pointer](#)
- struct [is\\_polymorphic](#)
- struct [is\\_reference](#)
- struct [is\\_rvalue\\_reference](#)
- struct [is\\_same](#)
- struct [is\\_scalar](#)
- struct [is\\_standard\\_layout](#)
- struct [is\\_swappable](#)
- struct [is\\_swappable\\_with](#)
- struct [is\\_trivial](#)
- struct [is\\_trivially\\_assignable](#)
- struct [is\\_trivially\\_constructible](#)
- struct [is\\_trivially\\_default\\_constructible](#)
- struct [is\\_trivially\\_destructible](#)
- struct [is\\_union](#)
- struct [is\\_void](#)
- struct [is\\_volatile](#)
- class [istream\\_iterator](#)
- class [istreambuf\\_iterator](#)
- struct [iterator](#)
- struct [iterator\\_traits< \\_Tp \\* >](#)
- struct [iterator\\_traits< const \\_Tp \\* >](#)
- class [length\\_error](#)
- struct [less](#)
- struct [less< void >](#)
- struct [less\\_equal](#)
- struct [less\\_equal< void >](#)
- class [linear\\_congruential\\_engine](#)
- class [list](#)
- class [locale](#)
- class [lock\\_guard](#)
- class [logic\\_error](#)
- struct [logical\\_and](#)
- struct [logical\\_and< void >](#)
- struct [logical\\_not](#)
- struct [logical\\_not< void >](#)
- struct [logical\\_or](#)
- struct [logical\\_or< void >](#)
- class [lognormal\\_distribution](#)
- struct [make\\_signed](#)
- struct [make\\_unsigned](#)
- class [map](#)
- class [mask\\_array](#)
- class [match\\_results](#)



- class [mem\\_fun1\\_ref\\_t](#)
- class [mem\\_fun1\\_t](#)
- class [mem\\_fun\\_ref\\_t](#)
- class [mem\\_fun\\_t](#)
- class [mersenne\\_twister\\_engine](#)
- class [messages](#)
- struct [messages\\_base](#)
- class [messages\\_byname](#)
- struct [minus](#)
- struct [minus< void >](#)
- struct [modulus](#)
- struct [modulus< void >](#)
- class [money\\_base](#)
- class [money\\_get](#)
- class [money\\_put](#)
- class [moneypunct](#)
- class [moneypunct\\_byname](#)
- class [move\\_iterator](#)
- class [multimap](#)
- struct [multiplies](#)
- struct [multiplies< void >](#)
- class [multiset](#)
- class [mutex](#)
- struct [negate](#)
- struct [negate< void >](#)
- class [negative\\_binomial\\_distribution](#)
- class [nested\\_exception](#)
- class [normal\\_distribution](#)
- struct [not\\_equal\\_to](#)
- struct [not\\_equal\\_to< void >](#)
- class [num\\_get](#)
- class [num\\_put](#)
- struct [numeric\\_limits](#)
- struct [numeric\\_limits< bool >](#)
- struct [numeric\\_limits< char >](#)
- struct [numeric\\_limits< char16\\_t >](#)
- struct [numeric\\_limits< char32\\_t >](#)
- struct [numeric\\_limits< double >](#)
- struct [numeric\\_limits< float >](#)
- struct [numeric\\_limits< int >](#)
- struct [numeric\\_limits< long >](#)
- struct [numeric\\_limits< long double >](#)
- struct [numeric\\_limits< long long >](#)
- struct [numeric\\_limits< short >](#)
- struct [numeric\\_limits< signed char >](#)
- struct [numeric\\_limits< unsigned char >](#)
- struct [numeric\\_limits< unsigned int >](#)
- struct [numeric\\_limits< unsigned long >](#)
- struct [numeric\\_limits< unsigned long long >](#)
- struct [numeric\\_limits< unsigned short >](#)
- struct [numeric\\_limits< wchar\\_t >](#)

- class `num_punct`
- class `num_punct_byname`
- struct `once_flag`
- class `ostream_iterator`
- class `ostreambuf_iterator`
- class `out_of_range`
- struct `output_iterator_tag`
- class `overflow_error`
- struct `owner_less`
- struct `owner_less< shared_ptr< _Tp > >`
- struct `owner_less< void >`
- struct `owner_less< weak_ptr< _Tp > >`
- class `packaged_task< _Res(_ArgTypes...)>`
- struct `pair`
- class `piecewise_constant_distribution`
- struct `piecewise_construct_t`
- class `piecewise_linear_distribution`
- struct `plus`
- class `pointer_to_binary_function`
- class `pointer_to_unary_function`
- struct `pointer_traits`
- struct `pointer_traits< _Tp * >`
- class `poisson_distribution`
- class `priority_queue`
- class `promise`
- class `promise< _Res & >`
- class `promise< void >`
- class `queue`
- struct `random_access_iterator_tag`
- class `random_device`
- class `range_error`
- struct `rank`
- struct `ratio`
- struct `ratio_equal`
- struct `ratio_not_equal`
- class `raw_storage_iterator`
- class `recursive_mutex`
- class `recursive_timed_mutex`
- class `reference_wrapper`
- class `regex_error`
- class `regex_iterator`
- class `regex_token_iterator`
- class `regex_traits`
- struct `remove_all_extents`
- struct `remove_const`
- struct `remove_cv`
- struct `remove_extent`
- struct `remove_pointer`
- struct `remove_reference`
- struct `remove_volatile`
- class `result_of`

- class [reverse\\_iterator](#)
- class [runtime\\_error](#)
- class [scoped\\_allocator\\_adaptor](#)
- class [seed\\_seq](#)
- class [set](#)
- class [shared\\_future](#)
- class [shared\\_future< \\_Res & >](#)
- class [shared\\_future< void >](#)
- class [shared\\_lock](#)
- class [shared\\_ptr](#)
- class [shared\\_timed\\_mutex](#)
- class [shuffle\\_order\\_engine](#)
- class [slice](#)
- class [slice\\_array](#)
- class [stack](#)
- class [student\\_t\\_distribution](#)
- class [sub\\_match](#)
- class [subtract\\_with\\_carry\\_engine](#)
- class [system\\_error](#)
- class [thread](#)
- class [time\\_base](#)
- class [time\\_get](#)
- class [time\\_get\\_byname](#)
- class [time\\_put](#)
- class [time\\_put\\_byname](#)
- class [timed\\_mutex](#)
- struct [try\\_to\\_lock\\_t](#)
- class [tuple](#)
- class [tuple< \\_T1, \\_T2 >](#)
- struct [tuple\\_element](#)
- struct [tuple\\_element< 0, std::pair< \\_Tp1, \\_Tp2 > >](#)
- struct [tuple\\_element< 0, tuple< \\_Head, \\_Tail... > >](#)
- struct [tuple\\_element< 1, std::pair< \\_Tp1, \\_Tp2 > >](#)
- struct [tuple\\_element< \\_\\_i, tuple< \\_Head, \\_Tail... > >](#)
- struct [tuple\\_element< \\_\\_i, tuple<> >](#)
- struct [tuple\\_element< \\_Int, ::array< \\_Tp, \\_Nm > >](#)
- struct [tuple\\_element< \\_Int, std::\\_\\_debug::array< \\_Tp, \\_Nm > >](#)
- struct [tuple\\_size](#)
- struct [tuple\\_size< std::\\_\\_debug::array< \\_Tp, \\_Nm > >](#)
- struct [tuple\\_size< std::pair< \\_Tp1, \\_Tp2 > >](#)
- struct [tuple\\_size< tuple< \\_Elements... > >](#)
- struct [tuple\\_size< ::array< \\_Tp, \\_Nm > >](#)
- struct [type\\_index](#)
- class [type\\_info](#)
- struct [unary\\_function](#)
- class [unary\\_negate](#)
- class [underflow\\_error](#)
- struct [underlying\\_type](#)
- class [uniform\\_int\\_distribution](#)
- class [uniform\\_real\\_distribution](#)
- class [unique\\_lock](#)

- class [unique\\_ptr](#)
- class [unique\\_ptr< \\_Tp\[\], \\_Dp >](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)
- struct [uses\\_allocator](#)
- struct [uses\\_allocator< tuple< \\_Types... >, \\_Alloc >](#)
- class [valarray](#)
- class [vector](#)
- class [vector< bool, \\_Alloc >](#)
- class [wbuffer\\_convert](#)
- class [weak\\_ptr](#)
- class [weibull\\_distribution](#)
- class [wstring\\_convert](#)

## Typedefs

- `template<typename _Alloc, typename _Up >`  
`using \_\_alloc\_rebind = typename __allocator_traits_base::template __rebind< _Alloc, _Up >::type`
- `template<typename _Tp >`  
`using \_\_allocator\_base = \_\_gnu\_cxx::new\_allocator< _Tp >`
- `template<typename _Fn, typename... _Args>`  
`using \_\_async\_result\_of = typename result\_of< typename decay< _Fn >::type(typename decay< _Args >::type...)>::type`
- `typedef unsigned char \_\_atomic\_flag\_data\_type`
- `template<bool __v>`  
`using \_\_bool\_constant = integral\_constant< bool, __v >`
- `typedef FILE \_\_c\_file`
- `typedef __locale_t \_\_c\_locale`
- `typedef __pthread_mutex_t \_\_c\_lock`
- `template<typename _Tp, typename _Hash >`  
`using \_\_cache\_default = __not_< __and_< __is_fast_hash< _Hash >, __is_nothrow_invocable< const _Hash &, const _Tp & > >>`
- `template<typename _Fn, typename... _Args>`  
`using \_\_call\_is\_nothrow = __call_is_nothrow< __invoke_result< _Fn, _Args... >, _Fn, _Args... >`
- `template<typename _From, typename _To >`  
`using \_\_check\_func\_return\_type = __or_< is\_void< _To >, is\_same< _From, _To >, is\_convertible< _From, _To > >`
- `typedef basic\_string< char > \_\_cow\_string`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using \_\_detected\_or = \_\_detector< _Default, void, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using \_\_detected\_or\_t = typename \_\_detected\_or< _Default, _Op, _Args... >::type`
- `template<typename _Tp >`  
`using \_\_do\_is\_convertible\_to\_basic\_istream\_impl = decltype(__is_convertible_to_basic_istream_↵  
test(declval< typename remove\_reference< _Tp >::type * >()))`
- `template<typename _Tp >`  
`using \_\_do\_is\_convertible\_to\_basic\_ostream\_impl = decltype(__is_convertible_to_basic_ostream_↵  
test(declval< typename remove\_reference< _Tp >::type * >()))`

- `template<typename _Tp >`  
`using __empty_not_final = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp >`  
`>::type`
- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up><`  
`::value>::type, size_t = tuple_size<_Tp>::value>`  
`using __enable_if_has_tuple_size = _Tp`
- `template<bool _Cond, typename _Tp = void>`  
`using __enable_if_t = typename enable_if< _Cond, _Tp >::type`
- `template<typename _Tp >`  
`using __get_first_arg_t = typename __get_first_arg< _Tp >::type`
- `template<typename _Alloc, typename _Tp >`  
`using __is_erased_or_convertible = __or_< is_same< _Tp, __erased_type >, is_convertible< _Alloc, _Tp >`  
`>`
- `template<typename _Tp, typename... _Args>`  
`using __is_nothrow_constructible_impl = __is_nt_constructible_impl< __is_constructible(_Tp, _Args...), _Tp,`  
`_Args... >`
- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>`  
`using __is_socketlike = __or_< is_integral< _Tp2 >, is_enum< _Tp2 > >`
- `template<typename _Tp >`  
`using __make_not_void = typename conditional< is_void< _Tp >::value, __undefined, _Tp >::type`
- `template<typename _Alloc >`  
`using __outer_allocator_t = decltype(std::declval< _Alloc >().outer_allocator())`
- `template<typename _Ptr, typename _Tp >`  
`using __ptr_rebind = typename pointer_traits< _Ptr >::template rebind< _Tp >`
- `template<typename _Tp, typename _Up >`  
`using __replace_first_arg_t = typename __replace_first_arg< _Tp, _Up >::type`
- `template<typename _Tp >`  
`using __rethrow_if_nested_cond = typename enable_if< __and_< is_polymorphic< _Tp >, __or_< __not_<`  
`is_base_of< nested_exception, _Tp >, is_convertible< _Tp *, nested_exception * > >>::value >::type`
- `template<typename _Istream >`  
`using __rvalue_istream_type = typename __is_convertible_to_basic_istream< _Istream >::__istream_type`
- `template<typename _Ostream >`  
`using __rvalue_ostream_type = typename __is_convertible_to_basic_ostream< _Ostream >::__ostream_type`
- `typedef basic_string< char > __sso_string`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`using __sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch<`  
`_alloc >`
- `template<std::size_t __i, typename _Tp >`  
`using __tuple_element_t = typename tuple_element< __i, _Tp >::type`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc`  
`= std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`  
`using __umap_hashtable = Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, <`  
`Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy,`  
`_Tr >`
- `template<bool _Cache>`  
`using __umap_traits = __detail::__Hashtable_traits< _Cache, false, true >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc`  
`= std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>`  
`using __ummap_hashtable = Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, <`  
`Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy,`  
`_Tr >`
- `template<bool _Cache>`  
`using __ummap_traits = __detail::__Hashtable_traits< _Cache, false, false >`

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>  
using __umset_hashtable = \_Hashtable< _Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using __umset_traits = \_\_detail::\_Hashtable\_traits< _Cache, true, false >`
- `template<typename _Tp, typename _Alloc, typename... _Args>  
using __uses_alloc_t = \_\_uses\_alloc< uses\_allocator< _Tp, _Alloc >::value, _Tp, _Alloc, _Args... >`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>  
using __uset_hashtable = \_Hashtable< _Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using __uset_traits = \_\_detail::\_Hashtable\_traits< _Cache, true, true >`
- `template<typename... >  
using __void_t = void`
- `typedef unsigned long _Bit_type`
- `template<typename... _Cond>  
using _Require = typename enable\_if< __and< _Cond... >::value >::type`
- `template<typename _Alloc >  
using _RequireAllocator = typename enable\_if< __is_allocator< _Alloc >::value, _Alloc >::type`
- `template<typename _InIter >  
using _RequireInputIter = typename enable\_if< is\_convertible< typename iterator_traits< _InIter >::iterator, __category, input\_iterator\_tag >::value >::type`
- `template<std::size_t __i, typename _Tuple >  
using _Safe_tuple_element_t = typename enable\_if< (__i < tuple\_size< _Tuple >::value), tuple\_element< __i, _Tuple > >::type::type`
- `template<typename _Tp >  
using add_const_t = typename add\_const< _Tp >::type`
- `template<typename _Tp >  
using add_cv_t = typename add\_cv< _Tp >::type`
- `template<typename _Tp >  
using add_lvalue_reference_t = typename add\_lvalue\_reference< _Tp >::type`
- `template<typename _Tp >  
using add_pointer_t = typename add\_pointer< _Tp >::type`
- `template<typename _Tp >  
using add_rvalue_reference_t = typename add\_rvalue\_reference< _Tp >::type`
- `template<typename _Tp >  
using add_volatile_t = typename add\_volatile< _Tp >::type`
- `template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>  
using aligned_storage_t = typename aligned\_storage< _Len, _Align >::type`
- `template<size_t _Len, typename... _Types>  
using aligned_union_t = typename aligned\_union< _Len, _Types... >::type`
- `typedef atomic< bool > atomic\_bool`
- `typedef atomic< char > atomic\_char`
- `typedef atomic< char16_t > atomic\_char16\_t`
- `typedef atomic< char32_t > atomic\_char32\_t`
- `typedef atomic< int > atomic\_int`
- `typedef atomic< int16_t > atomic\_int16\_t`
- `typedef atomic< int32_t > atomic\_int32\_t`
- `typedef atomic< int64_t > atomic\_int64\_t`
- `typedef atomic< int8_t > atomic\_int8\_t`

- typedef `atomic< int_fast16_t > atomic_int_fast16_t`
- typedef `atomic< int_fast32_t > atomic_int_fast32_t`
- typedef `atomic< int_fast64_t > atomic_int_fast64_t`
- typedef `atomic< int_fast8_t > atomic_int_fast8_t`
- typedef `atomic< int_least16_t > atomic_int_least16_t`
- typedef `atomic< int_least32_t > atomic_int_least32_t`
- typedef `atomic< int_least64_t > atomic_int_least64_t`
- typedef `atomic< int_least8_t > atomic_int_least8_t`
- typedef `atomic< intmax_t > atomic_intmax_t`
- typedef `atomic< intptr_t > atomic_intptr_t`
- typedef `atomic< long long > atomic_llong`
- typedef `atomic< long > atomic_long`
- typedef `atomic< ptrdiff_t > atomic_ptrdiff_t`
- typedef `atomic< signed char > atomic_schar`
- typedef `atomic< short > atomic_short`
- typedef `atomic< size_t > atomic_size_t`
- typedef `atomic< unsigned char > atomic_uchar`
- typedef `atomic< unsigned int > atomic_uint`
- typedef `atomic< uint16_t > atomic_uint16_t`
- typedef `atomic< uint32_t > atomic_uint32_t`
- typedef `atomic< uint64_t > atomic_uint64_t`
- typedef `atomic< uint8_t > atomic_uint8_t`
- typedef `atomic< uint_fast16_t > atomic_uint_fast16_t`
- typedef `atomic< uint_fast32_t > atomic_uint_fast32_t`
- typedef `atomic< uint_fast64_t > atomic_uint_fast64_t`
- typedef `atomic< uint_fast8_t > atomic_uint_fast8_t`
- typedef `atomic< uint_least16_t > atomic_uint_least16_t`
- typedef `atomic< uint_least32_t > atomic_uint_least32_t`
- typedef `atomic< uint_least64_t > atomic_uint_least64_t`
- typedef `atomic< uint_least8_t > atomic_uint_least8_t`
- typedef `atomic< uintmax_t > atomic_uintmax_t`
- typedef `atomic< uintptr_t > atomic_uintptr_t`
- typedef `atomic< unsigned long long > atomic_ullong`
- typedef `atomic< unsigned long > atomic_ulong`
- typedef `atomic< unsigned short > atomic_ushort`
- typedef `atomic< wchar_t > atomic_wchar_t`
- typedef `match_results< const char * > cmatch`
- template<typename... \_Tp>  
using `common_type_t` = typename `common_type< _Tp... >::type`
- template<bool \_Cond, typename \_Iftrue, typename \_Iffalse >  
using `conditional_t` = typename `conditional< _Cond, _Iftrue, _Iffalse >::type`
- typedef `regex_iterator< const char * > cregex_iterator`
- typedef `regex_token_iterator< const char * > cregex_token_iterator`
- typedef `sub_match< const char * > csub_match`
- template<typename \_Tp >  
using `decay_t` = typename `decay< _Tp >::type`
- typedef `minstd_rand0 default_random_engine`
- template<bool \_Cond, typename \_Tp = void>  
using `enable_if_t` = typename `enable_if< _Cond, _Tp >::type`
- typedef `integral_constant< bool, false > false_type`
- typedef `basic_filebuf< char > filebuf`

- typedef [basic\\_fstream](#)< char > [fstream](#)
- typedef [basic\\_ifstream](#)< char > [ifstream](#)
- template<size\_t... \_Idx>  
using [index\\_sequence](#) = [integer\\_sequence](#)< size\_t, \_Idx... >
- template<typename... \_Types>  
using [index\\_sequence\\_for](#) = [make\\_index\\_sequence](#)< sizeof...(\_Types)>
- typedef [basic\\_ios](#)< char > [ios](#)
- typedef [basic\\_iostream](#)< char > [iostream](#)
- typedef [basic\\_istream](#)< char > [istream](#)
- typedef [basic\\_istreamstream](#)< char > [istreamstream](#)
- typedef [shuffle\\_order\\_engine](#)< minstd\_rand0, 256 > [knuth\\_b](#)
- template<size\_t \_Num>  
using [make\\_index\\_sequence](#) = [make\\_integer\\_sequence](#)< size\_t, \_Num >
- template<typename \_Tp, \_Tp \_Num>  
using [make\\_integer\\_sequence](#) = [integer\\_sequence](#)< \_Tp, \_\_integer\_pack(\_Num)... >
- template<typename \_Tp >  
using [make\\_signed\\_t](#) = typename [make\\_signed](#)< \_Tp >::type
- template<typename \_Tp >  
using [make\\_unsigned\\_t](#) = typename [make\\_unsigned](#)< \_Tp >::type
- typedef enum [std::memory\\_order](#) [memory\\_order](#)
- typedef [linear\\_congruential\\_engine](#)< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > [minstd\\_rand](#)
- typedef [linear\\_congruential\\_engine](#)< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [minstd\\_rand0](#)
- typedef [mersenne\\_twister\\_engine](#)< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef [mersenne\\_twister\\_engine](#)< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > [mt19937\\_64](#)
- typedef void(\* [new\\_handler](#)) ()
- typedef [basic\\_ofstream](#)< char > [ofstream](#)
- typedef [basic\\_ostream](#)< char > [ostream](#)
- typedef [basic\\_ostreamstream](#)< char > [ostreamstream](#)
- typedef \_\_PTRDIFF\_TYPE\_\_ [ptrdiff\\_t](#)
- typedef [discard\\_block\\_engine](#)< [ranlux24\\_base](#), 223, 23 > [ranlux24](#)
- typedef [subtract\\_with\\_carry\\_engine](#)< uint\_fast32\_t, 24, 10, 24 > [ranlux24\\_base](#)
- typedef [discard\\_block\\_engine](#)< [ranlux48\\_base](#), 389, 11 > [ranlux48](#)
- typedef [subtract\\_with\\_carry\\_engine](#)< uint\_fast64\_t, 48, 5, 12 > [ranlux48\\_base](#)
- template<typename \_R1, typename \_R2 >  
using [ratio\\_divide](#) = typename \_\_ratio\_divide< \_R1, \_R2 >::type
- template<typename \_R1, typename \_R2 >  
using [ratio\\_multiply](#) = typename \_\_ratio\_multiply< \_R1, \_R2 >::type
- typedef [basic\\_regex](#)< char > [regex](#)
- template<typename \_Tp >  
using [remove\\_all\\_extents\\_t](#) = typename [remove\\_all\\_extents](#)< \_Tp >::type
- template<typename \_Tp >  
using [remove\\_const\\_t](#) = typename [remove\\_const](#)< \_Tp >::type
- template<typename \_Tp >  
using [remove\\_cv\\_t](#) = typename [remove\\_cv](#)< \_Tp >::type
- template<typename \_Tp >  
using [remove\\_extent\\_t](#) = typename [remove\\_extent](#)< \_Tp >::type
- template<typename \_Tp >  
using [remove\\_pointer\\_t](#) = typename [remove\\_pointer](#)< \_Tp >::type



---

- `template<typename _Tp >`  
`using remove\_reference\_t = typename remove\_reference< _Tp >::type`
- `template<typename _Tp >`  
`using remove\_volatile\_t = typename remove\_volatile< _Tp >::type`
- `template<typename _Tp >`  
`using result\_of\_t = typename result\_of< _Tp >::type`
- `typedef __SIZE_TYPE__ size\_t`
- `typedef match\_results< string::const_iterator > smatch`
- `typedef regex\_iterator< string::const_iterator > sregex\_iterator`
- `typedef regex\_token\_iterator< string::const_iterator > sregex\_token\_iterator`
- `typedef sub\_match< string::const_iterator > ssub\_match`
- `typedef basic\_streambuf< char > streambuf`
- `typedef long long streamoff`
- `typedef fpos< mbstate_t > streampos`
- `typedef ptrdiff_t streamsize`
- `typedef basic\_string< char > string`
- `typedef basic\_stringbuf< char > stringbuf`
- `typedef basic\_stringstream< char > stringstream`
- `typedef void(* terminate\_handler) ()`
- `typedef integral\_constant< bool, true > true\_type`
- `template<std::size_t __i, typename _Tp >`  
`using tuple\_element\_t = typename tuple\_element< __i, _Tp >::type`
- `typedef fpos< mbstate_t > u16streampos`
- `typedef basic\_string< char16_t > u16string`
- `typedef fpos< mbstate_t > u32streampos`
- `typedef basic\_string< char32_t > u32string`
- `template<typename _Tp >`  
`using underlying\_type\_t = typename underlying\_type< _Tp >::type`
- `typedef void(* unexpected\_handler) ()`
- `template<typename... >`  
`using void\_t = void`
- `typedef match\_results< const wchar_t * > wcmatch`
- `typedef regex\_iterator< const wchar_t * > wcregex\_iterator`
- `typedef regex\_token\_iterator< const wchar_t * > wcregex\_token\_iterator`
- `typedef sub\_match< const wchar_t * > wcsub\_match`
- `typedef basic\_filebuf< wchar_t > wfilebuf`
- `typedef basic\_fstream< wchar_t > wfstream`
- `typedef basic\_ifstream< wchar_t > wifstream`
- `typedef basic\_ios< wchar_t > wios`
- `typedef basic\_iostream< wchar_t > wiostream`
- `typedef basic\_istream< wchar_t > wistream`
- `typedef basic\_istreamstream< wchar_t > wistreamstream`
- `typedef basic\_ofstream< wchar_t > wofstream`
- `typedef basic\_ostream< wchar_t > wostream`
- `typedef basic\_ostreamstream< wchar_t > wostringstream`
- `typedef basic\_regex< wchar_t > wregex`
- `typedef match\_results< wstring::const_iterator > wsmatch`
- `typedef regex\_iterator< wstring::const_iterator > wsregex\_iterator`
- `typedef regex\_token\_iterator< wstring::const_iterator > wsregex\_token\_iterator`
- `typedef sub\_match< wstring::const_iterator > wssub\_match`
- `typedef basic\_streambuf< wchar_t > wstreambuf`

---

- typedef [fpos](#)< mbstate\_t > [wstreampos](#)
- typedef [basic\\_string](#)< wchar\_t > [wstring](#)
- typedef [basic\\_stringbuf](#)< wchar\_t > [wstringbuf](#)
- typedef [basic\\_stringstream](#)< wchar\_t > [wstringstream](#)

## Enumerations

- enum { [\\_S\\_threshold](#) }
- enum { [\\_S\\_chunk\\_size](#) }
- enum { [\\_S\\_word\\_bit](#) }
- enum [\\_\\_memory\\_order\\_modifier](#) { [\\_\\_memory\\_order\\_mask](#), [\\_\\_memory\\_order\\_modifier\\_mask](#), [\\_\\_memory\\_order\\_hle\\_acquire](#), [\\_\\_memory\\_order\\_hle\\_release](#) }
- enum [\\_ios\\_Fmtflags](#) { [\\_S\\_boolalpha](#), [\\_S\\_dec](#), [\\_S\\_fixed](#), [\\_S\\_hex](#), [\\_S\\_internal](#), [\\_S\\_left](#), [\\_S\\_oct](#), [\\_S\\_right](#), [\\_S\\_scientific](#), [\\_S\\_showbase](#), [\\_S\\_showpoint](#), [\\_S\\_showpos](#), [\\_S\\_skipws](#), [\\_S\\_unitbuf](#), [\\_S\\_uppercase](#), [\\_S\\_adjustfield](#), [\\_S\\_basefield](#), [\\_S\\_floatfield](#), [\\_S\\_ios\\_fmtflags\\_end](#), [\\_S\\_ios\\_fmtflags\\_max](#), [\\_S\\_ios\\_fmtflags\\_min](#) }
- enum [\\_ios\\_Iostate](#) { [\\_S\\_goodbit](#), [\\_S\\_badbit](#), [\\_S\\_eofbit](#), [\\_S\\_failbit](#), [\\_S\\_ios\\_iostate\\_end](#), [\\_S\\_ios\\_iostate\\_max](#), [\\_S\\_ios\\_iostate\\_min](#) }
- enum [\\_ios\\_Openmode](#) { [\\_S\\_app](#), [\\_S\\_ate](#), [\\_S\\_bin](#), [\\_S\\_in](#), [\\_S\\_out](#), [\\_S\\_trunc](#), [\\_S\\_ios\\_openmode\\_end](#), [\\_S\\_ios\\_openmode\\_max](#), [\\_S\\_ios\\_openmode\\_min](#) }
- enum [\\_ios\\_Seekdir](#) { [\\_S\\_beg](#), [\\_S\\_cur](#), [\\_S\\_end](#), [\\_S\\_ios\\_seekdir\\_end](#) }
- enum [\\_Manager\\_operation](#) { [\\_\\_get\\_type\\_info](#), [\\_\\_get\\_functor\\_ptr](#), [\\_\\_clone\\_functor](#), [\\_\\_destroy\\_functor](#) }
- enum [\\_Rb\\_tree\\_color](#) { [\\_S\\_red](#), [\\_S\\_black](#) }
- enum [codecvt\\_mode](#) { [consume\\_header](#), [generate\\_header](#), [little\\_endian](#) }
- enum [cv\\_status](#) { [no\\_timeout](#), [timeout](#) }
- enum [errc](#) { [address\\_family\\_not\\_supported](#), [address\\_in\\_use](#), [address\\_not\\_available](#), [already\\_connected](#), [argument\\_list\\_too\\_long](#), [argument\\_out\\_of\\_domain](#), [bad\\_address](#), [bad\\_file\\_descriptor](#), [broken\\_pipe](#), [connection\\_aborted](#), [connection\\_already\\_in\\_progress](#), [connection\\_refused](#), [connection\\_reset](#), [cross\\_device\\_link](#), [destination\\_address\\_required](#), [device\\_or\\_resource\\_busy](#), [directory\\_not\\_empty](#), [executable\\_format\\_error](#), [file\\_exists](#), [file\\_too\\_large](#), [filename\\_too\\_long](#), [function\\_not\\_supported](#), [host\\_unreachable](#), [illegal\\_byte\\_sequence](#), [inappropriate\\_io\\_control\\_operation](#), [interrupted](#), [invalid\\_argument](#), [invalid\\_seek](#), [io\\_error](#), [is\\_a\\_directory](#), [message\\_size](#), [network\\_down](#), [network\\_reset](#), [network\\_unreachable](#), [no\\_buffer\\_space](#), [no\\_child\\_process](#), [no\\_lock\\_available](#), [no\\_message](#), [no\\_protocol\\_option](#), [no\\_space\\_on\\_device](#), [no\\_such\\_device\\_or\\_address](#), [no\\_such\\_device](#), [no\\_such\\_file\\_or\\_directory](#), [no\\_such\\_process](#), [not\\_a\\_directory](#), [not\\_a\\_socket](#), [not\\_connected](#), [not\\_enough\\_memory](#), [operation\\_in\\_progress](#), [operation\\_not\\_permitted](#), [operation\\_not\\_supported](#), [operation\\_would\\_block](#), [permission\\_denied](#), [protocol\\_not\\_supported](#), [read\\_only\\_file\\_system](#), [resource\\_deadlock\\_would\\_occur](#), [resource\\_unavailable\\_try\\_again](#), [result\\_out\\_of\\_range](#), [timed\\_out](#), [too\\_many\\_files\\_open\\_in\\_system](#), [too\\_many\\_files\\_open](#), [too\\_many\\_links](#), [too\\_many\\_symbolic\\_link\\_levels](#), [wrong\\_protocol\\_type](#) }
- enum [float\\_denorm\\_style](#) { [denorm\\_indeterminate](#), [denorm\\_absent](#), [denorm\\_present](#) }
- enum [float\\_round\\_style](#) { [round\\_indeterminate](#), [round\\_toward\\_zero](#), [round\\_to\\_nearest](#), [round\\_toward\\_infinity](#), [round\\_toward\\_neg\\_infinity](#) }

- enum `future_errc` { `future_already_retrieved`, `promise_already_satisfied`, `no_state`, `broken_promise` }
- enum `future_status` { `ready`, `timeout`, `deferred` }
- enum `io_errc` { `stream` }
- enum `launch` { `async`, `deferred` }
- enum `memory_order` {  
    `memory_order_relaxed`, `memory_order_consume`, `memory_order_acquire`, `memory_order_release`,  
    `memory_order_acq_rel`, `memory_order_seq_cst` }
- enum `pointer_safety` { `relaxed`, `preferred`, `strict` }

## Functions

- template<typename \_CharT >  
    \_CharT \* **\_\_add\_grouping** (\_CharT \* \_\_s, \_CharT \_\_sep, const char \* \_\_gbeg, size\_t \_\_gsize, const \_CharT \* \_\_first, const \_CharT \* \_\_last)
- template<typename \_Tp >  
    constexpr \_Tp \* **\_\_addressof** (\_Tp & \_\_r) noexcept
- template<typename \_ForwardIterator, typename \_BinaryPredicate >  
    \_FowardIterator **\_\_adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_BinaryPredicate \_\_↔ binary\_pred)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Tp, typename \_Compare >  
    void **\_\_adjust\_heap** (\_RandomAccessIterator \_\_first, \_Distance \_\_holeIndex, \_Distance \_\_len, \_Tp \_\_value, ↔ \_Compare \_\_comp)
- template<typename \_InputIterator, typename \_Distance >  
    \_GLIBCXX14\_CONSTEXPR void **\_\_advance** (\_InputIterator & \_\_i, \_Distance \_\_n, `input_iterator_tag`)
- template<typename \_BidirectionalIterator, typename \_Distance >  
    \_GLIBCXX14\_CONSTEXPR void **\_\_advance** (\_BidirectionalIterator & \_\_i, \_Distance \_\_n, `bidirectional_iterator_tag`)
- template<typename \_RandomAccessIterator, typename \_Distance >  
    \_GLIBCXX14\_CONSTEXPR void **\_\_advance** (\_RandomAccessIterator & \_\_i, \_Distance \_\_n, `random_access_iterator_tag`)
- template<typename \_Alloc >  
    void **\_\_alloc\_on\_copy** (\_Alloc & \_\_one, const \_Alloc & \_\_two)
- template<typename \_Alloc >  
    \_Alloc **\_\_alloc\_on\_copy** (const \_Alloc & \_\_a)
- template<typename \_Alloc >  
    void **\_\_alloc\_on\_move** (\_Alloc & \_\_one, \_Alloc & \_\_two)
- template<typename \_Alloc >  
    void **\_\_alloc\_on\_swap** (\_Alloc & \_\_one, \_Alloc & \_\_two)
- template<typename \_Alloc >  
    **\_\_allocated\_ptr**< \_Alloc > **\_\_allocate\_guarded** (\_Alloc & \_\_a)
- template<typename \_Tp, \_Lock\_policy \_Lp, typename \_Alloc, typename... \_Args>  
    \_\_shared\_ptr< \_Tp, \_Lp > **\_\_allocate\_shared** (const \_Alloc & \_\_a, \_Args &&... \_\_args)
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void atomic\_thread\_fence(`memory_order` \_\_m) noexcept
- namespace cxx11 **\_\_attribute\_\_** ((\_\_abi\_tag\_\_("cxx11")))
- template<typename \_Fn, typename \_Tp, typename... \_Args>  
    constexpr bool **\_\_call\_is\_nt** (\_invoke\_memfun\_ref)
- template<typename \_Fn, typename \_Tp, typename... \_Args>  
    constexpr bool **\_\_call\_is\_nt** (\_invoke\_memfun\_deref)
- template<typename \_Fn, typename \_Tp >  
    constexpr bool **\_\_call\_is\_nt** (\_invoke\_memobj\_ref)
- template<typename \_Fn, typename \_Tp >  
    constexpr bool **\_\_call\_is\_nt** (\_invoke\_memobj\_deref)
- template<typename \_Fn, typename... \_Args>  
    constexpr bool **\_\_call\_is\_nt** (\_invoke\_other)

- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet *__f)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __↵  
chunk_size, _Compare __comp)`
- `constexpr memory_order __cmpexch_failure_order (memory_order __m) noexcept`
- `constexpr memory_order __cmpexch_failure_order2 (memory_order __m) noexcept`
- `template<typename _Tp >`  
`_Tp __complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc __attribute__((__unused__)), char *__out, const int __size  
__attribute__((__unused__)), const char *__fmt,...)`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`

- `template<>`  
`void __convert_to_v (const char *, double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, long double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, ostreambuf\_iterator< _CharT > >::type __copy_↵`  
`__move_a2 (_CharT * __first, _CharT * __last, ostreambuf\_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, ostreambuf\_iterator< _CharT > >::type __copy_↵`  
`__move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf\_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, _CharT * >::type __copy_move_a2 (istreambuf\_iterator<`  
`_CharT > __first, istreambuf\_iterator< _CharT > __last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, ostreambuf\_iterator< _CharT, char\_traits< _CharT`  
`> >::type __copy_move_a2 (_CharT *, _CharT *, ostreambuf\_iterator< _CharT, char\_traits< _CharT >`  
`>)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, ostreambuf\_iterator< _CharT, char\_traits< _CharT`  
`> >::type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf\_iterator< _CharT, char\_traits<`  
`_CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, _CharT * >::type __copy_move_a2 (istreambuf\_iterator<`  
`_CharT, char\_traits< _CharT > >, istreambuf\_iterator< _CharT, char\_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random\_access\_iterator\_tag)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs (basic\_streambuf< _CharT, _Traits > *__sbin, basic\_streambuf< _CharT, _↵`  
`_Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs_eof (basic\_streambuf< _CharT, _Traits > *, basic\_streambuf< _CharT, _Traits`  
`> *, bool &)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic\_streambuf< char > *__sbin, basic\_streambuf< char > *__sbout,`  
`bool &__ineof)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic\_streambuf< wchar_t > *__sbin, basic\_streambuf< wchar_t > *↵`  
`__sbout, bool &__ineof)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type __count_if (_InputIterator __first, _InputIterator __last, _↵`  
`Predicate __pred)`
- `template<typename _Signature, typename _Fn, typename _Alloc >`  
`static shared\_ptr< __future_base::__Task_state_base< _Signature > > __create_task_state (_Fn &&__fn,`  
`const _Alloc &__a)`

- constexpr size\_t **\_\_deque\_buf\_size** (size\_t \_\_size)
- template<typename \_InputIterator >  
\_GLIBCXX14\_CONSTEXPR iterator\_traits< \_InputIterator >::difference\_type **\_\_distance** (\_InputIterator \_\_first, \_InputIterator \_\_last, [input\\_iterator\\_tag](#))
- template<typename \_RandomAccessIterator >  
\_GLIBCXX14\_CONSTEXPR iterator\_traits< \_RandomAccessIterator >::difference\_type **\_\_distance** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, [random\\_access\\_iterator\\_tag](#))
- template<typename \_Alloc >  
void **\_\_do\_alloc\_on\_copy** (\_Alloc &\_\_one, const \_Alloc &\_\_two, [true\\_type](#))
- template<typename \_Alloc >  
void **\_\_do\_alloc\_on\_copy** (\_Alloc &, const \_Alloc &, [false\\_type](#))
- template<typename \_Alloc >  
void **\_\_do\_alloc\_on\_move** (\_Alloc &\_\_one, \_Alloc &\_\_two, [true\\_type](#))
- template<typename \_Alloc >  
void **\_\_do\_alloc\_on\_move** (\_Alloc &, \_Alloc &, [false\\_type](#))
- template<typename \_Alloc >  
void **\_\_do\_alloc\_on\_swap** (\_Alloc &\_\_one, \_Alloc &\_\_two, [true\\_type](#))
- template<typename \_Alloc >  
void **\_\_do\_alloc\_on\_swap** (\_Alloc &, \_Alloc &, [false\\_type](#))
- template<typename \_OutStr, typename \_InChar, typename \_Codecvt, typename \_State, typename \_Fn >  
bool **\_\_do\_str\_codecvt** (const \_InChar \*\_\_first, const \_InChar \*\_\_last, \_OutStr &\_\_outstr, const \_Codecvt &\_\_cvt, \_State &\_\_state, size\_t &\_\_count, \_Fn \_\_fn)
- template<typename \_I1, typename \_I2 >  
bool **\_\_equal4** (\_I1 \_\_first1, \_I1 \_\_last1, \_I2 \_\_first2, \_I2 \_\_last2)
- template<typename \_I1, typename \_I2, typename \_BinaryPredicate >  
bool **\_\_equal4** (\_I1 \_\_first1, \_I1 \_\_last1, \_I2 \_\_first2, \_I2 \_\_last2, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_I1, typename \_I2 >  
bool **\_\_equal\_aux** (\_I1 \_\_first1, \_I1 \_\_last1, \_I2 \_\_first2)
- template<typename \_ForwardIterator, typename \_Tp, typename \_CompareItTp, typename \_CompareTplt >  
[pair](#)< \_ForwardIterator, \_ForwardIterator > **\_\_equal\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_CompareItTp \_\_comp\_it\_val, \_CompareTplt \_\_comp\_val\_it)
- template<typename \_Tp, typename \_Up = \_Tp>  
\_Tp **\_\_exchange** (\_Tp &\_\_obj, \_Up &&\_\_new\_val)
- template<typename \_ForwardIterator, typename \_Tp >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<!\\_\\_is\\_scalar< \\_Tp >::\\_\\_value, void >::\\_\\_type](#) **\_\_fill\_a** (\_ForwardIterator \_\_first, [\\_ForwardIterator](#) \_\_last, const \_Tp &\_\_value)
- template<typename \_ForwardIterator, typename \_Tp >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_scalar< \\_Tp >::\\_\\_value, void >::\\_\\_type](#) **\_\_fill\_a** (\_ForwardIterator \_\_first, [\\_ForwardIterator](#) \_\_last, const \_Tp &\_\_value)
- template<typename \_Tp >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_byte< \\_Tp >::\\_\\_value, void >::\\_\\_type](#) **\_\_fill\_a** (\_Tp \*\_\_first, \_Tp \*\_\_last, const \_Tp &\_\_c)
- void **\_\_fill\_bvector** (\_Bit\_type \*\_\_v, unsigned int \_\_first, unsigned int \_\_last, bool \_\_x)
- template<typename \_OutputIterator, typename \_Size, typename \_Tp >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<!\\_\\_is\\_scalar< \\_Tp >::\\_\\_value, \\_OutputIterator >::\\_\\_type](#) **\_\_fill\_n\_a** (\_OutputIterator \_\_first, \_Size \_\_n, const \_Tp &\_\_value)
- template<typename \_OutputIterator, typename \_Size, typename \_Tp >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_scalar< \\_Tp >::\\_\\_value, \\_OutputIterator >::\\_\\_type](#) **\_\_fill\_n\_a** (\_OutputIterator \_\_first, \_Size \_\_n, const \_Tp &\_\_value)
- template<typename \_Size, typename \_Tp >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_byte< \\_Tp >::\\_\\_value, \\_Tp \\* >::\\_\\_type](#) **\_\_fill\_n\_a** (\_Tp \*\_\_first, \_Size \_\_n, const \_Tp &\_\_c)

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >`  
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator __find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iterator, typename _Predicate >`  
`_Iterator __find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance >`  
`_InputIterator __find_if_not_n (_InputIterator __first, _Distance & __len, _Predicate __pred)`
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement __gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _IntType, typename _UniformRandomBitGenerator >`  
`pair< _IntType, _IntType > __gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBitGenerator && __g)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & __get_helper (_Tuple_impl< __i, _Head, _Tail... > & __t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & __get_helper (const _Tuple_impl< __i, _Head, _Tail... > & __t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr _Head & __get_helper2 (_Tuple_impl< __i, _Head, _Tail... > & __t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr const _Head & __get_helper2 (const _Tuple_impl< __i, _Head, _Tail... > & __t) noexcept`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`size_t __iconv_adaptor (size_t (* __func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char ** __inbuf, size_t * __inbytes, char ** __outbuf, size_t * __outbytes)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool __includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void __inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT >`  
`int __int_to_char (_CharT * __bufend, _ValueT __v, const _CharT * __lit, ios_base::fmtflags __flags, bool __dec)`



- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void __introspect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`  
`constexpr _Up && __in fwd (typename remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result<_Callable, _Args...>::type __invoke (_Callable &&__fn, _Args &&... __args)`  
`noexcept(__is_nothrow_invocable<_Callable, _Args...>::value)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res __invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res __invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res __invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res __invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res __invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Ch, typename _Up >`  
`basic_istream<_Ch, _Up> & __is_convertible_to_basic_istream_test (basic_istream<_Ch, _Up> *)`
- `template<typename _Ch, typename _Up >`  
`basic_ostream<_Ch, _Up> & __is_convertible_to_basic_ostream_test (basic_ostream<_Ch, _Up> *)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool __is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool __is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool __is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator __is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter >`  
`constexpr iterator_traits<_Iter>::iterator_category __iterator_category (const _Iter &)`
- `template<typename _II1, typename _II2 >`  
`bool __lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool __lexicographical_compare_impl (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `constexpr int __lg (int __n)`
- `constexpr unsigned __lg (unsigned __n)`



- constexpr long **\_\_lg** (long \_\_n)
- constexpr unsigned long **\_\_lg** (unsigned long \_\_n)
- constexpr long long **\_\_lg** (long long \_\_n)
- constexpr unsigned long long **\_\_lg** (unsigned long long \_\_n)
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator **\_\_lower\_bound** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_RandomAccessIterator, typename \_Compare >  
void **\_\_make\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare &\_\_comp)
- template<typename \_Iterator, typename \_ReturnType = typename conditional<\_\_move\_if\_noexcept\_cond<typename iterator\_traits<\_Iterator>::value\_type>::value, \_Iterator, move\_iterator<\_Iterator>>::type>  
\_GLIBCXX17\_CONSTEXPR \_ReturnType **\_\_make\_move\_if\_noexcept\_iterator** (\_Iterator \_\_i)
- template<typename \_Tp, typename \_ReturnType = typename conditional<\_\_move\_if\_noexcept\_cond<\_Tp>::value, const \_Tp\*, move\_iterator<\_Tp\*>>::type>  
\_GLIBCXX17\_CONSTEXPR \_ReturnType **\_\_make\_move\_if\_noexcept\_iterator** (\_Tp \* \_\_i)
- template<typename \_Iterator >  
\_GLIBCXX17\_CONSTEXPR **reverse\_iterator**< \_Iterator > **\_\_make\_reverse\_iterator** (\_Iterator \_\_i)
- template<typename \_Tp, \_Lock\_policy \_Lp, typename... \_Args>  
\_shared\_ptr< \_Tp, \_Lp > **\_\_make\_shared** (\_Args &&... \_\_args)
- template<typename \_ForwardIterator, typename \_Compare >  
\_GLIBCXX14\_CONSTEXPR \_ForwardIterator **\_\_max\_element** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Compare \_\_comp)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_OutputIterator, typename \_Compare >  
\_OutputIterator **\_\_merge** (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_InputIterator2 \_\_last2, \_OutputIterator \_\_result, \_Compare \_\_comp)
- template<typename \_BidirectionalIterator, typename \_Distance, typename \_Pointer, typename \_Compare >  
void **\_\_merge\_adaptive** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_middle, \_BidirectionalIterator \_\_last, \_Distance \_\_len1, \_Distance \_\_len2, \_Pointer \_\_buffer, \_Distance \_\_buffer\_size, \_Compare \_\_comp)
- template<typename \_RandomAccessIterator1, typename \_RandomAccessIterator2, typename \_Distance, typename \_Compare >  
void **\_\_merge\_sort\_loop** (\_RandomAccessIterator1 \_\_first, \_RandomAccessIterator1 \_\_last, \_RandomAccessIterator2 \_\_result, \_Distance \_\_step\_size, \_Compare \_\_comp)
- template<typename \_RandomAccessIterator, typename \_Pointer, typename \_Compare >  
void **\_\_merge\_sort\_with\_buffer** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Pointer \_\_buffer, \_Compare \_\_comp)
- template<typename \_BidirectionalIterator, typename \_Distance, typename \_Compare >  
void **\_\_merge\_without\_buffer** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_middle, \_BidirectionalIterator \_\_last, \_Distance \_\_len1, \_Distance \_\_len2, \_Compare \_\_comp)
- template<typename \_ForwardIterator, typename \_Compare >  
\_GLIBCXX14\_CONSTEXPR \_ForwardIterator **\_\_min\_element** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Compare \_\_comp)
- template<typename \_ForwardIterator, typename \_Compare >  
\_GLIBCXX14\_CONSTEXPR **pair**< \_ForwardIterator, \_ForwardIterator > **\_\_minmax\_element** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Compare \_\_comp)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_BinaryPredicate >  
**pair**< \_InputIterator1, \_InputIterator2 > **\_\_mismatch** (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_BinaryPredicate >  
**pair**< \_InputIterator1, \_InputIterator2 > **\_\_mismatch** (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_InputIterator2 \_\_last2, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_Iterator >  
\_Iterator **\_\_miter\_base** (\_Iterator \_\_it)

- `template<typename _Iterator >`  
`auto __miter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__miter_base(__it.base())))`
- `template<typename _Iterator >`  
`auto __miter_base (move_iterator< _Iterator > __it) -> decltype(__miter_base(__it.base()))`
- `template<typename _Iterator, typename _Compare >`  
`void __move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`  
`void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool __next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Iterator >`  
`_Iterator __niter_base (_Iterator __it)`
- `template<typename _Iterator >`  
`auto __niter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__niter_base(__it.base())))`
- `template<typename _Iterator, typename _Container >`  
`_Iterator __niter_base (__gnu_cxx::__normal_iterator< _Iterator, _Container > __it)`
- `template<typename _Iterator >`  
`auto __niter_base (move_iterator< _Iterator > __it) -> decltype(make_move_iterator(__niter_base(__it.base())))`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _Alloc >`  
`__outermost_type< _Alloc >::type & __outermost (_Alloc &__a)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator __partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare &__comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool __prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void __push_heap ( _RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value,`  
`_Compare &__comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator __remove_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _↵`  
`Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __remove_if ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator __replace_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _↵`  
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _Ex >`  
`__rethrow_if_nested_cond< _Ex > __rethrow_if_nested_impl (const _Ex *__ptr)`
- `void __rethrow_if_nested_impl (const void *)`
- `template<typename _BidirectionalIterator >`  
`void __reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`void __reverse ( _RandomAccessIterator __first, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`  
`_BidirectionalIterator1 __rotate_adaptive ( _BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _↵`  
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance _↵`  
`__buffer_size)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator >`  
`_RandomAccessIterator __sample ( _InputIterator __first, _InputIterator __last, input\_iterator\_tag, _Random↵`  
`AccessIterator __out, random\_access\_iterator\_tag, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↵`  
`Generator >`  
`_OutputIterator __sample ( _ForwardIterator __first, _ForwardIterator __last, forward\_iterator\_tag, _OutputIterator`  
`__out, _Cat, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 __search ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`  
`_ForwardIterator __search_n ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, _Unary↵`  
`Predicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`  
`_ForwardIterator __search_n_aux ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, _Unary↵`  
`Predicate __unary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate >`  
`_RandomAccessIter __search_n_aux ( _RandomAccessIter __first, _RandomAccessIter __last, _Integer __↵`  
`count, _UnaryPredicate __unary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __set_intersection ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __set_union ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵`  
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`  
`_ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`  
`void __stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_in (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_in (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_out (const _CharT * __first, const _CharT * __last, basic\_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_out (const _CharT * __first, const _CharT * __last, basic\_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `void __throw_bad_alloc (void) __attribute__((__noreturn__))`
- `void __throw_bad_cast (void) __attribute__((__noreturn__))`
- `void __throw_bad_exception (void) __attribute__((__noreturn__))`
- `void __throw_bad_function_call () __attribute__((__noreturn__))`
- `void __throw_bad_typeid (void) __attribute__((__noreturn__))`
- `void __throw_bad_weak_ptr ()`
- `void __throw_domain_error (const char *) __attribute__((__noreturn__))`
- `void __throw_future_error (int) __attribute__((__noreturn__))`
- `void __throw_invalid_argument (const char *) __attribute__((__noreturn__))`
- `void __throw_ios_failure (const char *) __attribute__((__noreturn__))`
- `void __throw_length_error (const char *) __attribute__((__noreturn__))`
- `void __throw_logic_error (const char *) __attribute__((__noreturn__))`
- `void __throw_out_of_range (const char *) __attribute__((__noreturn__))`
- `void __throw_out_of_range_fmt (const char *,...) __attribute__((__noreturn__)) __attribute__((__format__(_gnu_printf_)))`
- `void __throw_overflow_error (const char *) __attribute__((__noreturn__))`
- `void __throw_range_error (const char *) __attribute__((__noreturn__))`
- `void __throw_regex_error (regex\_constants::error\_type __ecode)`
- `void __throw_regex_error (regex\_constants::error\_type __ecode, const char * __what)`
- `void __throw_runtime_error (const char *) __attribute__((__noreturn__))`
- `void __throw_system_error (int) __attribute__((__noreturn__))`
- `void __throw_underflow_error (const char *) __attribute__((__noreturn__))`
- `template<typename _Tp >`  
`void __throw_with_nested_impl (_Tp &&__t, true\_type)`
- `template<typename _Tp >`  
`void __throw_with_nested_impl (_Tp &&__t, false\_type)`

- `template<typename _Tp >`  
`constexpr _Tp * __to_address (_Tp * __ptr) noexcept`
- `template<typename _Ptr >`  
`constexpr std::pointer\_traits< _Ptr >::element_type * __to_address (const _Ptr & __ptr)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare & __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Pointer, typename _ForwardIterator >`  
`void __uninitialized_construct_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_copy_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator __uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator __uninitialized_copy_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`pair< _InputIterator, _ForwardIterator > __uninitialized_copy_n_pair (_InputIterator __first, _Size __n, & __result, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`pair< _RandomAccessIterator, _ForwardIterator > __uninitialized_copy_n_pair (_RandomAccessIterator __first, _Size __n, & __result, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`pair< _InputIterator, _ForwardIterator > __uninitialized_copy_n_pair (_InputIterator __first, _Size __n, & __result, input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void __uninitialized_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`  
`_ForwardIterator __uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`  
`_ForwardIterator __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`_ForwardIterator __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocator< _Tp > &)`

- `template<typename _ForwardIterator >`  
`void __uninitialized_default_novalue (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size >`  
`_ForwardIterator __uninitialized_default_novalue_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`  
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`  
`_ForwardIterator __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`  
`_ForwardIterator __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator __unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator __upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`__uses_alloc_t< _Tp, _Alloc, _Args... > __use_alloc (const _Alloc &__a)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __use_alloc (const _Alloc &&)=delete`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __uses_allocator_construct (const _Alloc &__a, _Tp *__ptr, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`void __uses_allocator_construct_impl (__uses_alloc0 __a, _Tp *__ptr, _Args &&... __args)`

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __uses_allocator_construct_impl (__uses_alloc1< _Alloc > __a, _Tp * __ptr, _Args &&... __args)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __uses_allocator_construct_impl (__uses_alloc2< _Alloc > __a, _Tp * __ptr, _Args &&... __args)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __a, size_t __n, _Tp * __restrict __b)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __a, size_t __n, size_t __s, _Tp * __restrict __b)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __a, _Tp * __restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __src, size_t __n, size_t __s1, _Tp * __restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __a, const size_t * __restrict __i, _Tp * __restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __a, size_t __n, _Tp * __restrict __b, const size_t * __restrict __i)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp * __restrict __src, size_t __n, const size_t * __restrict __i, _Tp * __restrict __dst, const size_t * __restrict __j)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`



- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void *__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict __valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`



- `template<std::size_t _Ind, typename... _Tp>`  
`auto __volget (volatile tuple< _Tp... > &__tuple) -> __tuple_element_t< _Ind, tuple< _Tp... >> volatile &`
- `template<std::size_t _Ind, typename... _Tp>`  
`auto __volget (const volatile tuple< _Tp... > &__tuple) -> __tuple_element_t< _Ind, tuple< _Tp... >> const volatile &`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s, const _CharT * __ws, int __len)`
- `template<typename _CharT, typename _Outlter >`  
`_Outlter __write (_Outlter __s, const _CharT * __ws, int __len)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > & __e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > & __e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > & __e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > & __e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, const Expr< _Dom, _Tp > & __e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t`  
`__n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _T1, typename... _Args>`  
`void _Construct (_T1 *__p, _Args &&... __args)`
- `template<typename _T1 >`  
`void _Construct_novalue (_T1 *__p)`
- `template<typename _Tp >`  
`void _Destroy (_Tp *__pointer)`

- `template<typename _ForwardIterator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &)`
- `template<typename _ForwardIterator, typename _Size >`  
`_ForwardIterator _Destroy_n (_ForwardIterator __first, _Size __count)`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()`
- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_decrement (const _Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_increment (const _Rb_tree_node_base *__x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `void abort (void) throw ()`
- `long abs (long __i)`
- `long long abs (long long __x)`
- `template<typename _Tp >`  
`_Tp abs (const complex<_Tp> &)`
- `constexpr double abs (double __x)`
- `constexpr float abs (float __x)`
- `constexpr long double abs (long double __x)`
- `template<class _Dom >`  
`_Expr<_UnClos<_Abs, _Expr, _Dom>, typename _Dom::value_type> abs (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp >`  
`_Expr<_UnClos<_Abs, _ValArray, _Tp>, _Tp> abs (const valarray<_Tp> &__v)`
- `template<typename _InputIterator, typename _Tp >`  
`_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `constexpr float acos (float __x)`
- `constexpr long double acos (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if<__is_integer<_Tp>::__value, double>::__type acos (_Tp __x)`
- `template<class _Dom >`  
`_Expr<_UnClos<_Acos, _Expr, _Dom>, typename _Dom::value_type> acos (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp >`  
`_Expr<_UnClos<_Acos, _ValArray, _Tp>, _Tp> acos (const valarray<_Tp> &__v)`
- `template<typename _Tp >`  
`std::complex<_Tp> acos (const std::complex<_Tp> &__z)`
- `template<typename _Tp >`  
`std::complex<_Tp> acosh (const std::complex<_Tp> &__z)`
- `template<typename _Tp >`  
`_GLIBCXX17_CONSTEXPR _Tp * addressof (_Tp &__r) noexcept`

- `template<typename _Tp >`  
`const _Tp * addressof (const _Tp &&)=delete`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↵ BinaryOperation __binary_op)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_↵ pred)`
- `template<typename _InputIterator, typename _Distance >`  
`_GLIBCXX17_CONSTEXPR void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _CharT, typename _Distance >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type advance (istreambuf_iterator< _↵ CharT > &__i, _Distance __n)`
- `void * align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept`
- `template<typename _Iter, typename _Predicate >`  
`bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _Tp > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Iter, typename _Predicate >`  
`bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `constexpr float asin (float __x)`
- `constexpr long double asin (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type asin (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin (const _Expr< _Dom, typename ↵ _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`std::complex< _Tp > asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`



- long double [assoc\\_laguerrel](#) (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [assoc\\_legendre](#) (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- float [assoc\\_legendref](#) (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- long double [assoc\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- template<typename \_Fn, typename... \_Args>  
[future](#)< \_\_async\_result\_of< \_Fn, \_Args... > > [async](#) (launch \_\_policy, \_Fn &&\_\_fn, \_Args &&... \_\_args)
- template<typename \_Fn, typename... \_Args>  
[future](#)< \_\_async\_result\_of< \_Fn, \_Args... > > [async](#) (\_Fn &&\_\_fn, \_Args &&... \_\_args)
- constexpr float [atan](#) (float \_\_x)
- constexpr long double [atan](#) (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type [atan](#) (\_Tp \_\_x)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Atan, \_ValArray, \_Tp >, \_Tp > [atan](#) (const [valarray](#)< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Atan, \_Expr, \_Dom >, typename \_Dom::value\_type > [atan](#) (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
[std::complex](#)< \_Tp > [atan](#) (const [std::complex](#)< \_Tp > &\_\_z)
- constexpr float [atan2](#) (float \_\_y, float \_\_x)
- constexpr long double [atan2](#) (long double \_\_y, long double \_\_x)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type [atan2](#) (\_Tp \_\_y, \_Up \_\_x)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_Atan2, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_Dom1::value\_type > [atan2](#) (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_e1, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_e2)
- template<class \_Dom >  
\_Expr< \_BinClos< \_Atan2, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_Dom::value\_type > [atan2](#) (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const [valarray](#)< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_Atan2, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_Dom::value\_type > [atan2](#) (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_Atan2, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_Dom::value\_type > [atan2](#) (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_BinClos< \_Atan2, \_ValArray, \_ValArray, \_Tp, \_Tp >, \_Tp > [atan2](#) (const [valarray](#)< \_Tp > &\_\_v, const [valarray](#)< \_Tp > &\_\_w)
- template<typename \_Tp >  
\_Expr< \_BinClos< \_Atan2, \_ValArray, \_Constant, \_Tp, \_Tp >, \_Tp > [atan2](#) (const [valarray](#)< \_Tp > &\_\_v, const \_Tp &\_\_t)
- template<typename \_Tp >  
\_Expr< \_BinClos< \_Atan2, \_Constant, \_ValArray, \_Tp, \_Tp >, \_Tp > [atan2](#) (const \_Tp &\_\_t, const [valarray](#)< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_Atan2, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_Dom::value\_type > [atan2](#) (const [valarray](#)< typename \_Dom::valarray > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)

- `template<typename _Tp >`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &__z)`
- `int atexit (void(*) (void)) throw ()`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_add_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`

- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * atomic_fetch_sub_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor ( __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit ( __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `void atomic_flag_clear (atomic_flag *__a) noexcept`
- `void atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void atomic_init (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void atomic_init (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`bool atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`

- `template<typename _ITp >`  
`bool atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load_explicit (const atomic< _ITp > *__a, memory\_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load_explicit (const volatile atomic< _ITp > *__a, memory\_order __m) noexcept`
- `template<typename _ITp >`  
`void atomic_store (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void atomic_store (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void atomic_store_explicit (atomic< _ITp > *__a, _ITp __i, memory\_order __m) noexcept`
- `template<typename _ITp >`  
`void atomic_store_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory\_order __m) noexcept`
- `template<typename _Container >`  
`back\_insert\_iterator< _Container > back_inserter (_Container &__x)`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Tp, size_t _Nm>`  
`_GLIBCXX14_CONSTEXPR _Tp * begin (_Tp(&__arr)[_Nm])`
- `template<class _Tp >`  
`constexpr const _Tp * begin (initializer\_list< _Tp > __ils) noexcept`
- `template<class _Tp >`  
`_Tp * begin (valarray< _Tp > &__va)`
- `template<class _Tp >`  
`const _Tp * begin (const valarray< _Tp > &__va)`
- `template<typename _Tpa, typename _Tpb >`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type beta (_Tpa __a, _Tpb __b)`
- `float betaf (float __a, float __b)`
- `long double betal (long double __a, long double __b)`
- `template<typename _Filter, typename _Tp >`  
`bool binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`bool binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Func, typename... _BoundArgs>`  
`_Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`_Bindres_helper< _Result, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Operation, typename _Tp >`  
`binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp >`  
`binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)`

- `ios_base & boolalpha (ios_base & __base)`
- `template<typename _Container >  
constexpr auto cbegin (const _Container & __cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(__cont))`
- `constexpr float ceil (float __x)`
- `constexpr long double ceil (long double __x)`
- `template<typename _Tp >  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil (_Tp __x)`
- `template<typename _Container >  
constexpr auto cend (const _Container & __cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(__cont))`
- `template<typename _Tp >  
__gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)`
- `float comp_ellint_1f (float __k)`
- `long double comp_ellint_1l (long double __k)`
- `template<typename _Tp >  
__gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)`
- `float comp_ellint_2f (float __k)`
- `long double comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tp1 >  
__gnu_cxx::__promote_2< _Tp, _Tp1 >::__type comp_ellint_3 (_Tp __k, _Tp1 __nu)`
- `float comp_ellint_3f (float __k, float __nu)`
- `long double comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tp >  
complex< _Tp > conj (const complex< _Tp > &)`
- `template<typename _Tp >  
std::complex< typename __gnu_cxx::__promote< _Tp >::__type > conj (_Tp __x)`
- `template<typename _Tp, typename _Up >  
shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > & __r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp >  
__shared_ptr< _Tp, _Lp > const_pointer_cast (const __shared_ptr< _Tp1, _Lp > & __r) noexcept`
- `template<typename _Iter, typename _OIter >  
_OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`
- `template<typename _Tp >  
_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI >  
_OI copy (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >  
_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Blter1, typename _Blter2 >  
_Blter2 copy_backward (_Blter1, _Blter1, _Blter2)`
- `template<typename _Tp >  
_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >  
_BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * >`  
`> __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * >`  
`__result)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _Size, typename _OIter >`  
`_OIter copy_n (_Iter, _Size, _OIter)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _Tp >`  
`complex< _Tp > cos (const complex< _Tp > &)`
- `constexpr float cos (float __x)`
- `constexpr long double cos (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cos (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > cos (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`complex< _Tp > cosh (const complex< _Tp > &)`
- `constexpr float cosh (float __x)`
- `constexpr long double cosh (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cosh (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type > cosh (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp > &__v)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type count (_InputIterator __first, _InputIterator __last, const _Tp &←`  
`__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __first, _InputIterator __last, _Predicate`  
`__pred)`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto crbegin (const _Container &__cont) -> decltype(std::rbegin(__cont))`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto crend (const _Container &__cont) -> decltype(std::rend(__cont))`
- `exception_ptr current_exception () noexcept`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`

- long double [cyl\\_bessel\\_il](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [cyl\\_bessel\\_j](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [cyl\\_bessel\\_jf](#) (float \_\_nu, float \_\_x)
- long double [cyl\\_bessel\\_jl](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [cyl\\_bessel\\_k](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [cyl\\_bessel\\_kf](#) (float \_\_nu, float \_\_x)
- long double [cyl\\_bessel\\_kl](#) (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type [cyl\\_neumann](#) (\_Tpnu \_\_nu, \_Tp \_\_x)
- float [cyl\\_neumannf](#) (float \_\_nu, float \_\_x)
- long double [cyl\\_neumannl](#) (long double \_\_nu, long double \_\_x)
- [ios\\_base](#) & [dec](#) ([ios\\_base](#) & \_\_base)
- void [declare\\_no\\_pointers](#) (char \*, size\_t)
- void [declare\\_reachable](#) (void \*)
- template<typename \_Tp >  
auto [declval](#) () noexcept -> decltype(\_\_declval< \_Tp >())
- [ios\\_base](#) & [defaultfloat](#) ([ios\\_base](#) & \_\_base)
- template<typename \_InputIterator >  
\_\_GLIBCXX17\_CONSTEXPR iterator\_traits< \_InputIterator >::difference\_type [distance](#) (\_InputIterator \_\_first, ↔  
\_InputIterator \_\_last)
- template<typename \_Tp, typename \_Up >  
[shared\\_ptr](#)< \_Tp > [dynamic\\_pointer\\_cast](#) (const [shared\\_ptr](#)< \_Up > &\_\_r) noexcept
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr< \_Tp, \_Lp > [dynamic\\_pointer\\_cast](#) (const \_\_shared\_ptr< \_Tp1, \_Lp > &\_\_r) noexcept
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type [ellint\\_1](#) (\_Tp \_\_k, \_Tpp \_\_phi)
- float [ellint\\_1f](#) (float \_\_k, float \_\_phi)
- long double [ellint\\_1l](#) (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type [ellint\\_2](#) (\_Tp \_\_k, \_Tpp \_\_phi)
- float [ellint\\_2f](#) (float \_\_k, float \_\_phi)
- long double [ellint\\_2l](#) (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type [ellint\\_3](#) (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float [ellint\\_3f](#) (float \_\_k, float \_\_nu, float \_\_phi)
- long double [ellint\\_3l](#) (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Container >  
\_\_GLIBCXX17\_CONSTEXPR auto [end](#) (\_Container &\_\_cont) -> decltype(\_\_cont.end())
- template<typename \_Container >  
\_\_GLIBCXX17\_CONSTEXPR auto [end](#) (const \_Container &\_\_cont) -> decltype(\_\_cont.end())
- template<typename \_Tp, size\_t \_Nm>  
\_\_GLIBCXX14\_CONSTEXPR \_Tp \* [end](#) (\_Tp(&\_\_arr)[\_Nm])
- template<class \_Tp >  
constexpr const \_Tp \* [end](#) ([initializer\\_list](#)< \_Tp > \_\_ils) noexcept
- template<class \_Tp >  
\_Tp \* [end](#) ([valarray](#)< \_Tp > &\_\_va)
- template<class \_Tp >  
const \_Tp \* [end](#) (const [valarray](#)< \_Tp > &\_\_va)
- template<typename \_CharT, typename \_Traits >  
[basic\\_ostream](#)< \_CharT, \_Traits > & [endl](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > &ends (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2 >`  
`bool equal (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2 >`  
`bool equal (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _Tp, typename _Up = _Tp>`  
`_Tp exchange (_Tp &__obj, _Up && __new_val)`
- `void exit (int) throw ()`
- `template<typename _Tp >`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `constexpr float exp (float __x)`
- `constexpr long double exp (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp (_Tp __x)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp > & __v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > exp (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)`
- `float expintf (float __x)`
- `long double expintl (long double __x)`
- `constexpr float fabs (float __x)`
- `constexpr long double fabs (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type fabs (_Tp __x)`
- `template<typename _Tp >`  
`_Tp fabs (const std::complex< _Tp > & __z)`
- `template<typename _Filter, typename _Tp >`  
`void fill (_Filter, _Filter, const _Tp &)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool & __x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`



- `template<typename _Tp >`  
`void fill (const \_Deque\_iterator< _Tp, _Tp &, _Tp * > &__first, const \_Deque\_iterator< _Tp, _Tp &, _Tp * >`  
`&__last, const _Tp &__value)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf\_iterator< _CharT > >::__type find`  
`(istreambuf\_iterator< _CharT > __first, istreambuf\_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵`  
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵`  
`_ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _Forward↵`  
`Iterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _Forward↵`  
`Iterator __last2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios_base & fixed (ios\_base &__base)`
- `constexpr float floor (float __x)`
- `constexpr long double floor (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type floor (_Tp __x)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & flush (basic\_ostream< _CharT, _Traits > &__os)`
- `constexpr float fmod (float __x, float __y)`
- `constexpr long double fmod (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fmod (_Tp __x, _Up __y)`

- `template<typename _Iter, typename _Funct >`  
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`  
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp >`  
`constexpr _Tp && forward (typename std::remove\_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && forward (typename std::remove\_reference< _Tp >::type &&__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > forward_as_tuple (_Elements &&... __args) noexcept`
- `float frexp (float __x, int *__exp)`
- `long double frexp (long double __x, int *__exp)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type frexp (_Tp __x, int *__exp)`
- `template<typename _Container >`  
`front\_insert\_iterator< _Container > front_inserter (_Container &__x)`
- `const error_category & future_category () noexcept`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`  
`_RealType generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr tuple\_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (std::pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr tuple\_element< _Int, std::pair< _Tp1, _Tp2 > >::type && get (std::pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr const tuple\_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (const std::pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`  
`constexpr const tuple\_element< _Int, std::pair< _Tp1, _Tp2 > >::type && get (const std::pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp & get (pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp & get (const pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp && get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp && get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp & get (pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp & get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp && get (pair< _Up, _Tp > &&__p) noexcept`

- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp && get (const pair< _Up, _Tp > &&__p) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp & get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp && get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp & get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp && get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & get (tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & get (const tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && get (tuple< _Elements... > &&__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && get (const tuple< _Elements... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp & get (tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp && get (tuple< _Types... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp & get (const tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp && get (const tuple< _Types... > &&__t) noexcept`
- **Catalogs & get\_catalogs ()**
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _Del, typename _Tp >`  
`_Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)`
- `new_handler get_new_handler ()` noexcept
- `pointer_safety get_pointer_safety ()` noexcept
- `template<typename _Tp >`  
`pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept`
- `terminate_handler get_terminate ()` noexcept
- `template<typename _CharT >`  
`_Get_time< _CharT > get_time (std::tm *__tmb, const _CharT *__fmt)`
- `unexpected_handler get_unexpected ()` noexcept
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str)`
- `template<>`  
`basic_istream< char > & getline (basic_istream< char > & __in, basic_string< char > & __str, char __delim)`
- `template<>`  
`basic_istream< wchar_t > & getline (basic_istream< wchar_t > & __in, basic_string< wchar_t > & __str, wchar_t __delim)`
- `template<typename _Facet >`  
`bool has_facet (const locale & __loc) throw ()`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `ios_base & hex (ios_base & __base)`
- `ios_base & hexfloat (ios_base & __base)`
- `template<typename _Tp >`  
`constexpr _Tp imag (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type imag (_Tp)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _BIter >`  
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > inserter (_Container & __x, _Iterator __i)`

- `ios_base & internal (ios_base &__base)`
- `const error_category & iostream_category () noexcept`
- `template<typename _ForwardIterator, typename _Tp >  
void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter >  
bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >  
bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >  
bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >  
bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >  
_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >  
_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >  
_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >  
_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _Predicate >  
bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >  
bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter1, typename _Filter2 >  
bool is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >  
bool is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >  
bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >  
bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >  
bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >  
bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _Filter >  
bool is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >  
bool is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >  
bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >  
bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Filter >  
_Filter is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >  
_Filter is_sorted_until (_Filter, _Filter, _Compare)`

- `template<typename _ForwardIterator >`  
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT >`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Filter1, typename _Filter2 >`  
`void iter_swap (_Filter1, _Filter2)`
- `template<typename _Tp >`  
`_Tp kill_dependency (_Tp __y) noexcept`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)`
- `float laguerref (unsigned int __n, float __x)`
- `long double laguerrel (unsigned int __n, long double __x)`
- `constexpr float ldexp (float __x, int __exp)`
- `constexpr long double ldexp (long double __x, int __exp)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __l, _Tp __x)`
- `float legendref (unsigned int __l, float __x)`
- `long double legendrel (unsigned int __l, long double __x)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`

- `template<typename _I1, typename _I2 >`  
`bool lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare >`  
`bool lexicographical\_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _Tp >`  
`complex<_Tp> log (const complex<_Tp> &)`
- `constexpr float log (float __x)`
- `constexpr long double log (long double __x)`
- `template<typename _Tp >`  
`constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double>::\_\_type log (_Tp __x)`
- `template<typename _Tp >`  
`_Expr<_UnClos<_Log, _ValArray, _Tp>, _Tp> log (const valarray<_Tp> &__v)`
- `template<class _Dom >`  
`_Expr<_UnClos<_Log, _Expr, _Dom>, typename _Dom::value_type> log (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp >`  
`complex<_Tp> log10 (const complex<_Tp> &)`
- `constexpr float log10 (float __x)`
- `constexpr long double log10 (long double __x)`
- `template<typename _Tp >`  
`constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double>::\_\_type log10 (_Tp __x)`
- `template<class _Dom >`  
`_Expr<_UnClos<_Log10, _Expr, _Dom>, typename _Dom::value_type> log10 (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp >`  
`_Expr<_UnClos<_Log10, _ValArray, _Tp>, _Tp> log10 (const valarray<_Tp> &__v)`
- `template<typename _Filter, typename _Tp >`  
`_Filter lower\_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter lower\_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `error\_code make\_error\_code (future\_errc __errc) noexcept`
- `error\_code make\_error\_code (errc __e) noexcept`
- `error\_code make\_error\_code (io\_errc __e) noexcept`
- `error\_condition make\_error\_condition (future\_errc __errc) noexcept`
- `error\_condition make\_error\_condition (io\_errc __e) noexcept`
- `error\_condition make\_error\_condition (errc __e) noexcept`
- `template<typename _Ex >`  
`exception\_ptr make\_exception\_ptr (_Ex __ex) noexcept`
- `template<typename _RandomAccessIterator >`  
`void make\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter >`  
`void make\_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void make\_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void make\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR move\_iterator< _Iterator > make_move_iterator (_Iterator __i)`
- `template<typename _T1, typename _T2 >`  
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type`  
`> make_pair (_T1 && __x, _T2 && __y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse\_iterator< _Iterator > make_reverse_iterator (_Iterator __i)`
- `template<typename _Tp, typename... _Args>`  
`shared\_ptr< _Tp > make_shared (_Args &&... __args)`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::__type... > make_tuple (_Elements &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`_MakeUniq< _Tp >::__single_object make_unique (_Args &&... __args)`
- `template<typename _Tp >`  
`_MakeUniq< _Tp >::__array make_unique (size_t __num)`
- `template<typename _Tp, typename... _Args>`  
`_MakeUniq< _Tp >::__invalid_type make_unique (_Args &&...)=delete`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & max (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR _Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_GLIBCXX14_CONSTEXPR _Filter max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last,`  
`_Compare __comp)`
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::* __pm) noexcept`
- `template<typename _Ret, typename _Tp >`  
`mem\_fun\_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp >`  
`const\_mem\_fun\_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem\_fun1\_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const\_mem\_fun1\_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Ret, typename _Tp >`  
`mem\_fun\_ref\_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp >`  
`const\_mem\_fun\_ref\_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const\_mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg) const)`



- `void * memchr (void *__s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >  
_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >  
_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >  
_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >  
_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
__last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >  
_GLIBCXX14_CONSTEXPR const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >  
_GLIBCXX14_CONSTEXPR const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >  
_GLIBCXX14_CONSTEXPR _Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >  
_GLIBCXX14_CONSTEXPR _Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >  
_GLIBCXX14_CONSTEXPR _Filter min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >  
_GLIBCXX14_CONSTEXPR _Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >  
_GLIBCXX14_CONSTEXPR _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >  
_GLIBCXX14_CONSTEXPR _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, ↔  
_Compare __comp)`
- `template<typename _Tp >  
_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >  
_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b,  
_Compare __comp)`
- `template<typename _Tp >  
_GLIBCXX14_CONSTEXPR pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >  
_GLIBCXX14_CONSTEXPR pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >  
_GLIBCXX14_CONSTEXPR pair< _Filter, _Filter > minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >  
_GLIBCXX14_CONSTEXPR pair< _Filter, _Filter > minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >  
_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator ↔  
__first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >  
_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator ↔  
__first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >  
pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >  
pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`

- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `float modf (float __x, float *__iptr)`
- `long double modf (long double __x, long double *__iptr)`
- `template<typename _Tp >`  
`constexpr std::remove_reference< _Tp >::type && move ( _Tp && __t) noexcept`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move ( _Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI >`  
`_OI move ( _II __first, _II __last, _OI __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move ( _Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward ( _Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 move_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward ( _Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type move_if_noexcept ( _Tp & __x) noexcept`
- `template<typename _InputIterator >`  
`_GLIBCXX17_CONSTEXPR _InputIterator next ( _InputIterator __x, typename iterator_traits< _InputIterator >::difference_type __n=1)`
- `template<typename _Blter >`  
`bool next_permutation ( _Blter, _Blter)`
- `template<typename _Blter, typename _Compare >`  
`bool next_permutation ( _Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`bool next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `ios_base & noboolalpha (ios_base & __base)`
- `template<typename _Iter, typename _Predicate >`  
`bool none_of ( _Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool none_of ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _Tp >`  
`_Tp norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type norm (_Tp __x)`
- `ios_base & noshowbase (ios_base & __base)`
- `ios_base & noshowpoint (ios_base & __base)`
- `ios_base & noshowpos (ios_base & __base)`
- `ios_base & noskipws (ios_base & __base)`
- `template<typename _Predicate >`  
`_GLIBCXX14_CONSTEXPR unary_negate< _Predicate > not1 (const _Predicate & __pred)`
- `template<typename _Predicate >`  
`_GLIBCXX14_CONSTEXPR binary_negate< _Predicate > not2 (const _Predicate & __pred)`
- `void notify_all_at_thread_exit (condition_variable &, unique_lock< mutex >)`
- `ios_base & nouitbuf (ios_base & __base)`
- `ios_base & nouppercase (ios_base & __base)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __↵  
last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __↵  
last, _Compare __comp)`
- `ios_base & oct (ios_base & __base)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`  
`bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp, _↵  
CharT, _Traits, _Dist > & __y)`
- `template<typename _T1, typename _T2 >`  
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _Tp >`  
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &) noexcept`
- `template<typename _CharT, typename _Traits >`  
`bool operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits  
> & __b)`
- `template<typename _StateT >`  
`bool operator!= (const fpos< _StateT > & __lhs, const fpos< _StateT > & __rhs)`
- `template<typename _Tp, std::size_t _Nm >`  
`bool operator!= (const array< _Tp, _Nm > & __one, const array< _Tp, _Nm > & __two)`
- `template<typename _Tp >`  
`bool operator!= (const _Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator< _Tp > & __y) noexcept`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > & __x, const _Deque_iterator< _Tp, _Ref, _Ptr >  
& __y) noexcept`
- `bool operator!= (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > & __x, const _Deque_iterator< _Tp, _RefR, _PtrR  
> & __y) noexcept`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator!= (const reverse_iterator< _Iterator > & __x, const reverse_iterator<  
_Iterator > & __y)`

- `bool operator!= (const error\_code &__lhs, const error\_code &__rhs) noexcept`
- `bool operator!= (const error\_code &__lhs, const error\_condition &__rhs) noexcept`
- `bool operator!= (const error\_condition &__lhs, const error\_code &__rhs) noexcept`
- `template<typename _Tp, typename _Seq >  
bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `bool operator!= (const error\_condition &__lhs, const error\_condition &__rhs) noexcept`
- `template<typename _Val >  
bool operator!= (const \_List\_iterator< _Val > &__x, const \_List\_const\_iterator< _Val > &__y) noexcept`
- `template<typename _IteratorL, typename _IteratorR >  
\_GLIBCXX17\_CONSTEXPR bool operator!= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Seq >  
bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>  
bool operator!= (const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _Tp, typename _Up >  
bool operator!= (const shared\_ptr< _Tp > &__a, const shared\_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >  
bool operator!= (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >  
bool operator!= (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _Val >  
bool operator!= (const \_Rb\_tree\_iterator< _Val > &__x, const \_Rb\_tree\_const\_iterator< _Val > &__y) noexcept`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __not_equal_to, typename _Dom1::value_type >::result_type > operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >  
_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >  
_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _T1, typename _T2 >  
constexpr bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _OutA1, typename _OutA2, typename... _InA>  
bool operator!= (const scoped\_allocator\_adaptor< _OutA1, _InA... > &__a, const scoped\_allocator\_adaptor< _OutA2, _InA... > &__b) noexcept`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>  
bool operator!= (const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const`  
`std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`  
`bool operator!= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const`  
`std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_`  
`_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator!= (const move_iterator< _IteratorL > &__x, const move_iterator<`  
`_IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator!= (const move_iterator< _Iterator > &__x, const move_iterator<`   
`_Iterator > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_`  
`alloc > &__rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp`  
`>::result_type > operator!= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp`  
`>::result_type > operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp`  
`>::result_type > operator!= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const`  
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`

- `template<typename _Bi_iter >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator!= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _Bi_iter, class _Alloc >`  
`bool operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _RealType >`  
`bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _RealType >`  
`bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`



- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc> &__lhs, const _CharT *__rhs)`
- `template<class _Dom>`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __modulus, typename _Dom1::value_type>::result_type> operator% (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename __fun< __modulus, typename _Dom::value_type>::result_type> operator% (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp>, typename __fun< __modulus, _Tp>::result_type> operator% (const valarray< _Tp> &__v, const valarray< _Tp> &__w)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp>, typename __fun< __modulus, _Tp>::result_type> operator% (const valarray< _Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp>, typename __fun< __modulus, _Tp>::result_type> operator% (const _Tp &__t, const valarray< _Tp> &__v)`
- `constexpr memory_order operator& (memory_order __m, __memory_order_modifier __mod)`
- `constexpr _ios_Fmtflags operator& (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode operator& (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr launch operator& (launch __x, launch __y)`
- `constexpr _ios_ostate operator& (_ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2>, typename __fun< __bitwise_and, typename _Dom1::value_type>::result_type> operator& (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename __fun< __bitwise_and, typename _Dom::value_type>::result_type> operator& (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename __fun< __bitwise_and, typename _Dom::value_type>::result_type> operator& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`



- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_and, typename _Dom::value_type >::result_type > operator& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp`  
`>::result_type > operator& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp`  
`>::result_type > operator& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp`  
`>::result_type > operator& (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > operator&& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_and, typename`  
`_Dom1::value_type >::result_type > operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > operator&& (const typename _Dom::value_type`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > operator&& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > operator&& (const _Tp &__t, const valarray< _Tp > &__v)`
- `const _los_Fmtflags & operator&= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator&= (_los_Openmode &__a, _los_Openmode __b)`
- `launch & operator&= (launch &__x, launch __y)`
- `const _los_losestate & operator&= (_los_losestate &__a, _los_losestate __b)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __multiplies, typename`  
`_Dom1::value_type >::result_type > operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > operator* (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom, typename _Dom::`  
`value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom, typename _Dom::`  
`value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __`  
`multiplies, typename _Dom::value_type >::result_type > operator* (const valarray< typename _Dom::value`  
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::`  
`result_type > operator* (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::`  
`result_type > operator* (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::`  
`result_type > operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`Deque\_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const Deque\_iterator< _Tp, _Ref, _Ptr > &__x)`  
`noexcept`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse\_iterator< _Iterator > operator+ (typename reverse\_iterator< _Iterator >::`  
`difference_type __n, const reverse\_iterator< _Iterator > &__x)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __`  
`plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1::`  
`value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > operator+ (const valarray< typename _Dom::value_type > &__v,`  
`const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type >`  
`&__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type >`  
`&__e, const valarray< typename _Dom::value_type > &__v)`

- `template<typename _Tp >`  
`complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type >`  
`operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type >`  
`operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type >`  
`operator+ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type operator- (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`

- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >  
_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const _Deque_iterator< _Tp, _RefL, _PtrL >  
&__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _IteratorL, typename _IteratorR >  
_GLIBCXX17_CONSTEXPR auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator<  
_IteratorR > &__y) -> decltype(__y.base() - __x.base())`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1<←  
::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const  
_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >  
_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __←  
minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom::value_type &__t, const  
_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __←  
minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value←  
_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __←  
minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value←  
_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >  
_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __←  
minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename _Dom::value_type  
> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
complex< _Tp > operator- (const complex< _Tp > &__x)`
- `template<typename _Tp >  
_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type  
> operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type  
> operator- (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >  
_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type  
> operator- (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _IteratorL, typename _IteratorR >  
_GLIBCXX17_CONSTEXPR auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator< ←  
_IteratorR > &__y) -> decltype(__x.base() - __y.base())`
- `template<class _Dom >  
_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< ←  
__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom<←  
::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >  
_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __←  
divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom::value_type &__t, const  
_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< ←  
__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom<←  
::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename _Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename _Dom::value_type`  
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type`  
`> operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type`  
`> operator/ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type`  
`> operator/ (const _Tp &__t, const valarray< _Tp > &__v)`
- `bool operator< (const error_code &__lhs, const error_code &__rhs) noexcept`
- `template<typename _Tp , std::size_t _Nm>`  
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `bool operator< (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp , typename _Ref , typename _Ptr >`  
`bool operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr >`  
`&__y) noexcept`
- `template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >`  
`bool operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR`  
`> &__y) noexcept`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator<`  
`_Iterator > &__y)`
- `template<typename _Tp , typename _Seq >`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator<`  
`_IteratorR > &__y)`
- `template<typename _Tp , typename _Seq >`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp , typename _Up >`  
`bool operator< (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type >`  
`&__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > operator< (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type >`  
`&__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > operator< (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`bool operator< (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator< (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _T1, typename _T2 >`  
`constexpr bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator< (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator< (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator< (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`  
`bool operator< (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator< (const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub\_match< _Bi_`  
`__iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator<`  
`_IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator< (const move\_iterator< _Iterator > &__x, const move\_iterator<`   
`_Iterator > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool operator< (const sub\_match< _Bi_iter > &__lhs, const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_`  
`alloc > &__rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`operator< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`operator< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`operator< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Bi_iter >`  
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub\_match< _Bi_iter >`  
`&__rhs)`

- `template<typename _Bi_iter >`  
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator< (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`  
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)`



- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, __Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code`  
`&__e)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, __Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, __Put_money<`  
`_MoneyT > __f)`
- `template<class _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, __Put_time< _↵`  
`CharT > __f)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< _↵`  
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom, typename _Dom↵`  
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const typename _Dom::value_type &↵`  
`__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_left, typename ↵`  
`_Dom1::value_type >::result_type > operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom, typename ↵`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< ↵`  
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const valarray< typename _Dom::value↵`  
`_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _U↵`  
`IntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr<`  
`_Tp, _Lp > &__p)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex<`  
`_Tp > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _Ostream, typename _Tp >`  
`enable_if< __and< __not< is_lvalue_reference< _Ostream >, __is_convertible_to_basic_ostream< ↵`  
`_Ostream >, __is_insertable< __rvalue_ostream_type< _Ostream >, const _Tp & >::value, __rvalue↵`  
`ostream_type< _Ostream > >::type operator<< (_Ostream &&__os, const _Tp &__x)`



- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result<__`  
`_type > operator<< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result<__`  
`_type > operator<< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result<__`  
`_type > operator<< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`  
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const`  
`sub_match< _Bi_iter > &__m)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`chi_squared_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`gamma_distribution< _RealType > &__x)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const  
basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _Tp, std::size_t _Nm>  
bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator<= (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Ref, typename _Ptr >  
bool operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr >  
&__y) noexcept`
- `template<typename _Iterator >  
_GLIBCXX17_CONSTEXPR bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator<  
_Iterator > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >  
bool operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR >  
&__y) noexcept`
- `template<typename _Tp, typename _Seq >  
bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR >  
_GLIBCXX17_CONSTEXPR bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator<  
_IteratorR > &__y)`
- `template<typename _Tp, typename _Seq >  
bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`

- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const typename _Dom::value_type`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less_equal, typename`  
`_Dom1::value_type >::result_type > operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Up >`  
`bool operator<= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _T1, typename _T2 >`  
`constexpr bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`  
`bool operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator<`  
`_IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator<= (const move_iterator< _Iterator > &__x, const move_iterator<`  
`_Iterator > &__y)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >↵`  
`::result_type > operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less_equal, _Tp >↵`  
`::result_type > operator<= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >↵`  
`::result_type > operator<= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↵`  
`_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`*__rhs)`
- `template<typename _Bi_iter >`  
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator<= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc`  
`> &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`  
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`&__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key,`  
`_Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _T1, typename _T2 >`  
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _Tp >`  
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &) noexcept`
- `template<typename _CharT, typename _Traits >`  
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _StateT >`  
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Tp, std::size_t _Nm >`  
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator== (const Deque_iterator< _Tp, _Ref, _Ptr > &__x, const Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp >`  
`bool operator== (const Fwd_list_iterator< _Tp > &__x, const Fwd_list_const_iterator< _Tp > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator== (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `bool operator== (thread::id __x, thread::id __y) noexcept`
- `bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Val >`  
`bool operator== (const List_iterator< _Val > &__x, const List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Val >`  
`bool operator== (const Rb_tree_iterator< _Val > &__x, const Rb_tree_const_iterator< _Val > &__y) noexcept`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __equal_to, typename _Dom1::value_type >::result_type > operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool operator== (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator== (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`  
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< __Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_t >`  
`__type > operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type >`  
`operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type >`  
`operator== (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`  
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator== (const sub\_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`  
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator== (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >`  
`bool operator== (const sub\_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _RealType >`  
`bool operator== (const std::normal\_distribution< _RealType > &__d1, const std::normal\_distribution< _RealType > &__d2)`
- `template<typename _Bi_iter, typename _Alloc >`  
`bool operator== (const match\_results< _Bi_iter, _Alloc > &__m1, const match\_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Tp, typename _Alloc >`  
`_GLIBCXX_END_NAMESPACE_CXX11 bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`



- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, bool >::type operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _Tp, std::size_t _Nm >`  
`bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator> (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater, typename _Dom1::value_type >::result_type > operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`



- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator>` (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- `template<typename _Tp, typename _Up >`  
`bool operator>` (const `shared_ptr`< \_Tp > &\_\_a, const `shared_ptr`< \_Up > &\_\_b) noexcept
- `template<typename _Tp >`  
`bool operator>` (const `shared_ptr`< \_Tp > &\_\_a, `nullptr_t`) noexcept
- `template<typename _Tp >`  
`bool operator>` (`nullptr_t`, const `shared_ptr`< \_Tp > &\_\_a) noexcept
- `template<typename _T1, typename _T2 >`  
`constexpr bool operator>` (const `pair`< \_T1, \_T2 > &\_\_x, const `pair`< \_T1, \_T2 > &\_\_y)
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator>` (const `unique_ptr`< \_Tp, \_Dp > &\_\_x, const `unique_ptr`< \_Up, \_Ep > &\_\_y)
- `template<typename _Tp, typename _Dp >`  
`bool operator>` (const `unique_ptr`< \_Tp, \_Dp > &\_\_x, `nullptr_t`)
- `template<typename _Tp, typename _Dp >`  
`bool operator>` (`nullptr_t`, const `unique_ptr`< \_Tp, \_Dp > &\_\_x)
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator>` (const `multiset`< \_Key, \_Compare, \_Alloc > &\_\_x, const `multiset`< \_Key, \_Compare, \_Alloc > &\_\_y)
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator>` (const `set`< \_Key, \_Compare, \_Alloc > &\_\_x, const `set`< \_Key, \_Compare, \_Alloc > &\_\_y)
- `template<typename _Bilter >`  
`bool operator>` (const `sub_match`< \_Bilter > &\_\_lhs, const `sub_match`< \_Bilter > &\_\_rhs)
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator>` (const `__sub_match_string`< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const `sub_match`< \_Bi\_iter > &\_\_rhs)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator>` (const `multimap`< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_x, const `multimap`< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_y)
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool operator>` (const `move_iterator`< \_IteratorL > &\_\_x, const `move_iterator`< \_IteratorR > &\_\_y)
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator>` (const `move_iterator`< \_Iterator > &\_\_x, const `move_iterator`< \_Iterator > &\_\_y)
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool operator>` (const `sub_match`< \_Bi\_iter > &\_\_lhs, const `__sub_match_string`< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_rhs)
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator>` (const \_Tp &\_\_t, const `valarray`< \_Tp > &\_\_v)
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator>` (const `valarray`< \_Tp > &\_\_v, const \_Tp &\_\_t)
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator>` (const `valarray`< \_Tp > &\_\_v, const `valarray`< \_Tp > &\_\_w)
- `template<typename _Bi_iter >`  
`bool operator>` (typename `iterator_traits`< \_Bi\_iter >::value\_type const \* \_\_lhs, const `sub_match`< \_Bi\_iter > &\_\_rhs)

- `template<typename _Bi_iter >`  
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bi_iter >`  
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator> (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >`  
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`bool operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator>= (thread::id __x, thread::id __y) noexcept`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<typename _IteratorL, typename _IteratorR >  
_GLIBCXX17_CONSTEXPR bool operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Seq >  
bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<class _Dom >  
_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >  
_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp, typename _Up >  
bool operator>= (const shared\_ptr< _Tp > &__a, const shared\_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >  
bool operator>= (const shared\_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >  
bool operator>= (nullptr_t, const shared\_ptr< _Tp > &__a) noexcept`
- `template<typename _T1, typename _T2 >  
constexpr bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >  
bool operator>= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >  
bool operator>= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >  
bool operator>= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >  
bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >  
bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >  
bool operator>= (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >  
bool operator>= (const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >  
bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _IteratorL, typename _IteratorR >  
_GLIBCXX17_CONSTEXPR bool operator>= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >  
_GLIBCXX17_CONSTEXPR bool operator>= (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >  
bool operator>= (const sub\_match< _Bi_iter > &__lhs, const \_\_sub\_match\_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >  
_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Bi_iter >  
bool operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >  
bool operator>= (const sub\_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >  
bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator>= (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename... _TElements, typename... _UElements >  
constexpr bool operator>= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >  
bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >  
bool operator>= (const sub\_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp >  
bool operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp >  
bool operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp >  
bool operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool operator>= (const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >  
bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >  
bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >  
bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_right, typename _Dom1::value_type >::result_type > operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, ValArray, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT > __f)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`

- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`  
`&__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _Istream, typename _Tp>`  
`enable_if< __and< __not< is_lvalue_reference< _Istream >, __is_convertible_to_basic_istream< _Istream >,`  
`__is_extractable< __rvalue_istream_type< _Istream >, _Tp && >::value, __rvalue_istream_type< _Istream >::`  
`type< _Istream >::type operator>> (_Istream &&__is, _Tp &&__x)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::`  
`result_type > operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::`  
`result_type > operator>> (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::`  
`result_type > operator>> (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution<`  
`_IntType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution<`  
`_RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`chi_squared_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`fisher_f_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`gamma_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string<`  
`_CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _↵`  
`_CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `constexpr _ios_Fmtflags operator^ ( _ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode operator^ ( _ios_Openmode __a, _ios_Openmode __b)`
- `constexpr launch operator^ (launch __x, launch __y)`
- `constexpr _ios_ostate operator^ ( _ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const valarray< typename _Dom↵`  
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename`  
`_Dom1::value_type >::result_type > operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`



- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::`  
`result_type > operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::`  
`result_type > operator^ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::`  
`result_type > operator^ (const valarray< _Tp > &__v, const _Tp &__t)`
- `const _los_Fmtflags & operator^= (_los_Fmtflags __a, _los_Fmtflags __b)`
- `const _los_Openmode & operator^= (_los_Openmode __a, _los_Openmode __b)`
- `launch & operator^= (launch __x, launch __y)`
- `const _los_losestate & operator^= (_los_losestate __a, _los_losestate __b)`
- `constexpr memory_order operator| (memory_order __m, __memory_order_modifier __mod)`
- `constexpr _los_Fmtflags operator| (_los_Fmtflags __a, _los_Fmtflags __b)`
- `constexpr _los_Openmode operator| (_los_Openmode __a, _los_Openmode __b)`
- `constexpr launch operator| (launch __x, launch __y)`
- `constexpr _los_losestate operator| (_los_losestate __a, _los_losestate __b)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename`  
`_Dom1::value_type >::result_type > operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::`  
`result_type > operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`



- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >↵`  
`::result_type > operator| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >↵`  
`::result_type > operator| (const _Tp &__t, const valarray< _Tp > &__v)`
- `const _los_Fmtflags & operator|= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator|= (_los_Openmode &__a, _los_Openmode __b)`
- `launch & operator|= (launch &__x, launch __y)`
- `const _los_losestate & operator|= (_los_losestate &__a, _los_losestate __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< ↵`  
`_logical_or, typename _Dom::value_type >::result_type > operator|| (const valarray< typename _Dom::value↵`  
`_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< ↵`  
`_logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom, typename _Dom↵`  
`::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< ↵`  
`_logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom, typename _Dom↵`  
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< ↵`  
`_logical_or, typename _Dom::value_type >::result_type > operator|| (const typename _Dom::value_type &↵`  
`__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename ↵`  
`_Dom1::value_type >::result_type > operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > operator|| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > operator|| (const _Tp &__t, const valarray< _Tp > &__v)`
- `constexpr _los_Fmtflags operator~ (_los_Fmtflags __a)`
- `constexpr _los_Openmode operator~ (_los_Openmode __a)`
- `constexpr launch operator~ (launch __x)`
- `constexpr _los_losestate operator~ (_los_losestate __a)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator ↵`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator ↵`  
`__last, _Compare __comp)`

- `template<typename _Iter, typename _RAIter >`  
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`  
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator >`  
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`void pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, int)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `constexpr float pow (float __x, float __y)`
- `constexpr long double pow (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`

- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type`  
`> pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_↵`  
`type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow (const _Tp &__t, const valarray< _Tp`  
`> &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const`  
`_Tp &__t)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > pow (const _Expr<`  
`_Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_↵`  
`_type > pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type`  
`> pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type`  
`> &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const`  
`valarray< _Tp > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_↵`  
`_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type`  
`&__t)`
- `template<typename _Tp , typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp >`  
`&__x, const _Up &__y)`
- `template<typename _Tp , typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const`  
`std::complex< _Up > &__y)`
- `template<typename _Tp , typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp >`  
`&__x, const std::complex< _Up > &__y)`
- `template<typename _BidirectionalIterator >`  
`_GLIBCXX17_CONSTEXPR _BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_traits< ↵`  
`_BidirectionalIterator >::difference_type __n=1)`
- `template<typename _Blter >`  
`bool prev_permutation (_Blter, _Blter)`
- `template<typename _Blter , typename _Compare >`  
`bool prev_permutation (_Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`bool prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator , typename _Compare >`  
`bool prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > proj (_Tp __x)`

- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > ptr_fun ( _Result(*__x)( _Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun ( _Result(*__x)( _Arg1, _Arg2))`
- `template<typename _RandomAccessIterator >`  
`void push_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void push_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`void push_heap ( _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void push_heap ( _RAIter, _RAIter, _Compare)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT >`  
`_Put_time< _CharT > put_time (const std::tm * __tmb, const _CharT * __fmt)`
- `template<typename _CharT >`  
`auto quoted (const _CharT * __string, _CharT __delim= _CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto quoted (const basic_string< _CharT, _Traits, _Alloc > & __string, _CharT __delim= _CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto quoted (basic_string< _CharT, _Traits, _Alloc > & __string, _CharT __delim= _CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _RAIter >`  
`void random_shuffle ( _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`  
`void random_shuffle ( _RAIter, _RAIter, _Generator &&)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void random_shuffle ( _RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto rbegin ( _Container & __cont) -> decltype(__cont.rbegin())`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto rbegin (const _Container & __cont) -> decltype(__cont.rbegin())`
- `template<typename _Tp, size_t _Nm >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Tp * > rbegin ( _Tp(& __arr)[ _Nm])`
- `template<typename _Tp >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< const _Tp * > rbegin (initializer_list< _Tp > __il)`
- `template<typename _Tp >`  
`constexpr _Tp real (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type real ( _Tp __x)`
- `template<typename _Filter, typename _Tp >`  
`_Filter remove ( _Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator remove ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`_OIter remove_copy ( _Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator remove_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`

- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto rend (_Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Container >`  
`_GLIBCXX17_CONSTEXPR auto rend (const _Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Tp * > rend (_Tp(&__arr)[_Nm])`
- `template<typename _Tp >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< const _Tp * > rend (initializer_list< _Tp > __il)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`_OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void rethrow_if_nested (const _Ex &__ex)`
- `template<typename _Tp >`  
`void return_temporary_buffer (_Tp * __p)`
- `template<typename _BIter >`  
`void reverse (_BIter, _BIter)`
- `template<typename _BidirectionalIterator >`  
`void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter, typename _OIter >`  
`_OIter reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __s)`

- float [riemann\\_zetaf](#) (float \_\_s)
- long double [riemann\\_zetal](#) (long double \_\_s)
- [ios\\_base](#) & [right](#) ([ios\\_base](#) & \_\_base)
- template<typename \_Filter, typename \_OIter >  
\_OIter [rotate\\_copy](#) (\_Filter, \_Filter, \_Filter, \_OIter)
- template<typename \_ForwardIterator, typename \_OutputIterator >  
\_OutputIterator [rotate\\_copy](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_middle, \_ForwardIterator \_\_last, \_↵  
OutputIterator \_\_result)
- [ios\\_base](#) & [scientific](#) ([ios\\_base](#) & \_\_base)
- template<typename \_Filter1, typename \_Filter2 >  
\_Filter1 [search](#) (\_Filter1, \_Filter1, \_Filter2, \_Filter2)
- template<typename \_Filter1, typename \_Filter2, typename \_BinaryPredicate >  
\_Filter1 [search](#) (\_Filter1, \_Filter1, \_Filter2, \_Filter2, \_BinaryPredicate)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2 >  
\_ForwardIterator1 [search](#) (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2, \_↵  
ForwardIterator2 \_\_last2)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2, typename \_BinaryPredicate >  
\_ForwardIterator1 [search](#) (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2, \_↵  
ForwardIterator2 \_\_last2, \_BinaryPredicate \_\_predicate)
- template<typename \_Filter, typename \_Size, typename \_Tp >  
\_Filter [search\\_n](#) (\_Filter, \_Filter, \_Size, const \_Tp &)
- template<typename \_Filter, typename \_Size, typename \_Tp, typename \_BinaryPredicate >  
\_Filter [search\\_n](#) (\_Filter, \_Filter, \_Size, const \_Tp &, \_BinaryPredicate)
- template<typename \_ForwardIterator, typename \_Integer, typename \_Tp >  
\_ForwardIterator [search\\_n](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Integer \_\_count, const \_Tp & \_\_val)
- template<typename \_ForwardIterator, typename \_Integer, typename \_Tp, typename \_BinaryPredicate >  
\_ForwardIterator [search\\_n](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Integer \_\_count, const \_Tp & \_\_val,  
\_BinaryPredicate \_\_binary\_pred)
- template<typename \_IIter1, typename \_IIter2, typename \_OIter >  
\_OIter [set\\_difference](#) (\_IIter1, \_IIter1, \_IIter2, \_IIter2, \_OIter)
- template<typename \_IIter1, typename \_IIter2, typename \_OIter, typename \_Compare >  
\_OIter [set\\_difference](#) (\_IIter1, \_IIter1, \_IIter2, \_IIter2, \_OIter, \_Compare)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_OutputIterator >  
\_OutputIterator [set\\_difference](#) (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_Input↵  
Iterator2 \_\_last2, \_OutputIterator \_\_result)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_OutputIterator, typename \_Compare >  
\_OutputIterator [set\\_difference](#) (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_Input↵  
Iterator2 \_\_last2, \_OutputIterator \_\_result, \_Compare \_\_comp)
- template<typename \_IIter1, typename \_IIter2, typename \_OIter >  
\_OIter [set\\_intersection](#) (\_IIter1, \_IIter1, \_IIter2, \_IIter2, \_OIter)
- template<typename \_IIter1, typename \_IIter2, typename \_OIter, typename \_Compare >  
\_OIter [set\\_intersection](#) (\_IIter1, \_IIter1, \_IIter2, \_IIter2, \_OIter, \_Compare)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_OutputIterator >  
\_OutputIterator [set\\_intersection](#) (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_Input↵  
Iterator2 \_\_last2, \_OutputIterator \_\_result)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_OutputIterator, typename \_Compare >  
\_OutputIterator [set\\_intersection](#) (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_Input↵  
Iterator2 \_\_last2, \_OutputIterator \_\_result, \_Compare \_\_comp)
- [new\\_handler](#) [set\\_new\\_handler](#) ([new\\_handler](#)) throw ()
- template<typename \_IIter1, typename \_IIter2, typename \_OIter >  
\_OIter [set\\_symmetric\\_difference](#) (\_IIter1, \_IIter1, \_IIter2, \_IIter2, \_OIter)
- template<typename \_IIter1, typename \_IIter2, typename \_OIter, typename \_Compare >  
\_OIter [set\\_symmetric\\_difference](#) (\_IIter1, \_IIter1, \_IIter2, \_IIter2, \_OIter, \_Compare)

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `terminate\_handler set\_terminate (terminate\_handler) noexcept`
- `unexpected\_handler set\_unexpected (unexpected\_handler) noexcept`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT >`  
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios\_base::fmtflags __mask)`
- `_Setprecision setprecision (int __n)`
- `_Setw setw (int __n)`
- `ios\_base & showbase (ios\_base & __base)`
- `ios\_base & showpoint (ios\_base & __base)`
- `ios\_base & showpos (ios\_base & __base)`
- `template<typename _RAIter, typename _UGenerator >`  
`void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`
- `template<typename _Tp >`  
`complex< _Tp > sin (const complex< _Tp > &)`
- `constexpr float sin (float __x)`
- `constexpr long double sin (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sin (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > sin (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp > & __v)`
- `template<typename _Tp >`  
`complex< _Tp > sinh (const complex< _Tp > &)`
- `constexpr float sinh (float __x)`
- `constexpr long double sinh (long double __x)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > sinh (const valarray< _Tp > & __v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > sinh (const _Expr< _Dom, typename _Dom::value_type > & __e)`

- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sinh (_Tp __x)`
- `ios_base & skipws (ios_base & __base)`
- `template<typename _RAlter >`  
`void sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`  
`void sort (_RAlter, _RAlter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAlter >`  
`void sort_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`  
`void sort_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`complex< _Tp > sqrt (const complex< _Tp > &)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp > & __v)`
- `constexpr float sqrt (float __x)`
- `constexpr long double sqrt (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sqrt (_Tp __x)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RAlter >`  
`void stable_sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`  
`void stable_sort (_RAlter, _RAlter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`



- `template<typename _RandomAccessIterator, typename _Compare >`  
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Up >`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `char * strchr (char * __s, int __n)`
- `char * strpbrk (char * __s1, const char * __s2)`
- `char * strrchr (char * __s, int __n)`
- `char * strstr (char * __s1, const char * __s2)`
- `void swap (_Bit_reference __x, _Bit_reference __y) noexcept`
- `void swap (_Bit_reference __x, bool &__y) noexcept`
- `void swap (bool &__x, _Bit_reference __y) noexcept`
- `void swap (thread &__x, thread &__y) noexcept`
- `template<typename _Tp, std::size_t _Nm>`  
`enable_if< ::_array_traits< _Tp, _Nm >::Is_swappable::value >::type swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`
- `template<typename _Tp, std::size_t _Nm>`  
`enable_if< !::_array_traits< _Tp, _Nm >::Is_swappable::value >::type swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, typename _Seq >`  
`enable_if< __is_swappable< _Seq >::value >::type swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Mutex >`  
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`enable_if< __is_swappable< _Seq >::value >::type swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp >`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _T1, typename _T2 >`  
`enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2 >`  
`enable_if< !__and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &, pair< _T1, _T2 > &)=delete`
- `template<typename _Tp >`  
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Dp >`  
`enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp, typename _Dp >`  
`enable_if< !__is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete`
- `template<typename _Tp, typename _Sequence, typename _Compare >`  
`enable_if< __and< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Res, typename... _Args>`  
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y)`

- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_istream< _CharT, _Traits, _Allocator > &__x, basic_istream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)`
- `template<typename _Res >`  
`void swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`
- `template<class _CharT, class _Traits >`  
`void swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Res, typename... _ArgTypes>`  
`void swap (packaged_task< _Res(_ArgTypes...) > &__x, packaged_task< _Res(_ArgTypes...) > &__y) noexcept`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename... _Elements>`  
`enable_if< __and< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`enable_if< !__and< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Bi_iter, typename _Alloc >`  
`void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Tp >`  
`enable_if< __and< __not< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > >::value >::type swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm>`  
`enable_if< __is_swappable< _Tp >::value >::type swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Tp >`  
`complex< _Tp > tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float tan (float __x)`
- `constexpr long double tan (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type tan (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > tanh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type > tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float tanh (float __x)`

- constexpr long double **tanh** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **tanh** (\_Tp \_\_x)
- void **terminate** () noexcept \_\_attribute\_\_((\_\_noreturn\_\_))
- template<typename \_Tp >  
void **throw\_with\_nested** (\_Tp &&\_\_t)
- template<typename... \_Elements>  
constexpr **tuple**< \_Elements &... > **tie** (\_Elements &... \_\_args) noexcept
- template<typename \_CharT >  
\_CharT **tolower** (\_CharT \_\_c, const **locale** &\_\_loc)
- template<typename \_CharT >  
\_CharT **toupper** (\_CharT \_\_c, const **locale** &\_\_loc)
- template<typename \_Iter, typename \_OIter, typename \_UnaryOperation >  
\_OIter **transform** (\_Iter, \_Iter, \_OIter, \_UnaryOperation)
- template<typename \_Iter1, typename \_Iter2, typename \_OIter, typename \_BinaryOperation >  
\_OIter **transform** (\_Iter1, \_Iter1, \_Iter2, \_OIter, \_BinaryOperation)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_UnaryOperation >  
\_OutputIterator **transform** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, \_UnaryOperation \_\_unary\_op)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_OutputIterator, typename \_BinaryOperation >  
\_OutputIterator **transform** (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_OutputIterator \_\_result, \_BinaryOperation \_\_binary\_op)
- template<typename... \_Tpls, typename = typename enable\_if<\_\_and< \_\_is\_tuple\_like< \_Tpls>...>::value>::type>  
constexpr auto **tuple\_cat** (\_Tpls &&... \_\_tpls) -> typename \_\_tuple\_cat\_result< \_Tpls... >::\_\_type
- \_GLIBCXX17\_DEPRECATED bool **uncaught\_exception** () noexcept \_\_attribute\_\_((\_\_pure\_\_))
- int **uncaught\_exceptions** () noexcept \_\_attribute\_\_((\_\_pure\_\_))
- void **undeclare\_no\_pointers** (char \*, size\_t)
- template<typename \_Tp >  
\_Tp \* **undeclare\_reachable** (\_Tp \*\_\_p)
- void **unexpected** () \_\_attribute\_\_((\_\_noreturn\_\_))
- template<typename \_InputIterator, typename \_ForwardIterator >  
\_ForwardIterator **uninitialized\_copy** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_ForwardIterator \_\_result)
- template<typename \_InputIterator, typename \_Size, typename \_ForwardIterator >  
\_ForwardIterator **uninitialized\_copy\_n** (\_InputIterator \_\_first, \_Size \_\_n, \_ForwardIterator \_\_result)
- template<typename \_ForwardIterator, typename \_Tp >  
void **uninitialized\_fill** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_x)
- template<typename \_ForwardIterator, typename \_Size, typename \_Tp >  
\_ForwardIterator **uninitialized\_fill\_n** (\_ForwardIterator \_\_first, \_Size \_\_n, const \_Tp &\_\_x)
- template<typename \_Filter >  
\_Filter **unique** (\_Filter, \_Filter)
- template<typename \_Filter, typename \_BinaryPredicate >  
\_Filter **unique** (\_Filter, \_Filter, \_BinaryPredicate)
- template<typename \_ForwardIterator >  
\_ForwardIterator **unique** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_ForwardIterator, typename \_BinaryPredicate >  
\_ForwardIterator **unique** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_Iter, typename \_OIter >  
\_OIter **unique\_copy** (\_Iter, \_Iter, \_OIter)
- template<typename \_Iter, typename \_OIter, typename \_BinaryPredicate >  
\_OIter **unique\_copy** (\_Iter, \_Iter, \_OIter, \_BinaryPredicate)
- template<typename \_InputIterator, typename \_OutputIterator >  
\_OutputIterator **unique\_copy** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result)

- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
  - `ios\_base & unitbuf (ios\_base & __base)`
  - `template<typename _Filter, typename _Tp >`  
`_Filter upper\_bound (_Filter, _Filter, const _Tp &)`
  - `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter upper\_bound (_Filter, _Filter, const _Tp &, _Compare)`
  - `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
  - `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
  - `ios\_base & uppercase (ios\_base & __base)`
  - `template<typename _Facet >`  
`const _Facet & use\_facet (const locale & __loc)`
  - `wchar_t * wcschr (wchar_t * __p, wchar_t __c)`
  - `wchar_t * wcsbrk (wchar_t * __s1, const wchar_t * __s2)`
  - `wchar_t * wcsrchr (wchar_t * __p, wchar_t __c)`
  - `wchar_t * wcsstr (wchar_t * __s1, const wchar_t * __s2)`
  - `wchar_t * wmemchr (wchar_t * __p, wchar_t __c, size_t __n)`
  - `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & ws (basic\_istream< _CharT, _Traits > & __is)`
- 
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
  - `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
  - `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
- 
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > & __is, bitset< _Nb > & __x)`
  - `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > & __os, const bitset< _Nb > & __x)`
- 
- `template<typename _Tp >`  
`complex< _Tp > operator+ (const complex< _Tp > & __x, const complex< _Tp > & __y)`
  - `template<typename _Tp >`  
`complex< _Tp > operator+ (const complex< _Tp > & __x, const _Tp & __y)`

- `template<typename _Tp >`  
`complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool operator!= (const complex< _Tp > &__x, const _Tp &__y)`

- `template<typename _Tp >`  
`constexpr bool operator!= (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
- `template<>`  
`basic_istream< char > & operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`

- `template<class _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > ref (_Tp &__t) noexcept`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > cref (const _Tp &__t) noexcept`
- `template<typename _Tp >`  
`void ref (const _Tp &&)=delete`
- `template<typename _Tp >`  
`void cref (const _Tp &&)=delete`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t) noexcept`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t) noexcept`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`  
`bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`  
`bool regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`



- `template<typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type`  
`__f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex<`  
`_Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_↵`  
`_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type,`  
`_Rx_traits > & __e, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↵`  
`_Rx_traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`  
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↵`  
`_Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, ↵`  
`s, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, const`  
`basic_regex< _Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type ↵`  
`__flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_↵`  
`traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type ↵`  
`__flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`  
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits >`  
`& __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > * __p)`
- `template<typename _Tp >`  
`bool atomic_is_lock_free (const shared_ptr< _Tp > * __p)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > * __p, memory_order)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > atomic_load (const shared_ptr< _Tp > * __p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic_load_explicit (const __shared_ptr< _Tp, _Lp > * __p, memory_order)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic\_load (const __shared_ptr< _Tp, _Lp > *__p)`
  
- `template<typename _Tp >`  
`void atomic\_store\_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory\_order)`
- `template<typename _Tp >`  
`void atomic\_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void atomic\_store\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void atomic\_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
  
- `template<typename _Tp >`  
`shared\_ptr< \_Tp > atomic\\_exchange\\_explicit \(shared\_ptr< \_Tp > \*\_\_p, shared\\_ptr< \\_Tp > \\_\\_r, memory\\\_order`
- `template<typename _Tp >`  
`shared\_ptr< \_Tp > atomic\\_exchange \(shared\_ptr< \_Tp > \*\_\_p, shared\\_ptr< \\_Tp > \\_\\_r\\)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic\_exchange\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic\_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
  
- `template<typename _Tp >`  
`bool atomic\_compare\_exchange\_strong\_explicit (shared_ptr< _Tp > *__p, shared\_ptr< \_Tp > \*\_\_v, shared\\_ptr< \\_Tp > \\_\\_w, memory\\\_order`
- `template<typename _Tp >`  
`bool atomic\_compare\_exchange\_strong (shared_ptr< _Tp > *__p, shared\_ptr< \_Tp > \*\_\_v, shared\\_ptr< \\_Tp > \\_\\_w\\)`
- `template<typename _Tp >`  
`bool atomic\_compare\_exchange\_weak\_explicit (shared_ptr< _Tp > *__p, shared\_ptr< \_Tp > \*\_\_v, shared\\_ptr< \\_Tp > \\_\\_w, memory\\\_order`
- `template<typename _Tp >`  
`bool atomic\_compare\_exchange\_weak (shared_ptr< _Tp > *__p, shared\_ptr< \_Tp > \*\_\_v, shared\\_ptr< \\_Tp > \\_\\_w\\)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_strong\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory\_order, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_weak\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory\_order __success, memory\_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`

## Variables

- static ios\_base::Init [\\_\\_ioinit](#)
- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
\_GLIBCXX17\_INLINE constexpr bool [\\_\\_is\\_nothrow\\_uses\\_allocator\\_constructible\\_v](#)
- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
\_GLIBCXX17\_INLINE constexpr bool [\\_\\_is\\_uses\\_allocator\\_constructible\\_v](#)
- \_GLIBCXX17\_INLINE constexpr [adopt\\_lock\\_t](#) [adopt\\_lock](#)
- \_GLIBCXX17\_INLINE constexpr [allocator\\_arg\\_t](#) [allocator\\_arg](#)
- \_GLIBCXX17\_INLINE constexpr [defer\\_lock\\_t](#) [defer\\_lock](#)
- \_GLIBCXX17\_INLINE constexpr \_Swallow\_assign [ignore](#)
- template<typename \_Tp >  
\_GLIBCXX17\_INLINE constexpr bool [is\\_nothrow\\_swappable\\_v](#)
- template<typename \_Tp, typename \_Up >  
\_GLIBCXX17\_INLINE constexpr bool [is\\_nothrow\\_swappable\\_with\\_v](#)
- template<typename \_Tp >  
\_GLIBCXX17\_INLINE constexpr bool [is\\_swappable\\_v](#)
- template<typename \_Tp, typename \_Up >  
\_GLIBCXX17\_INLINE constexpr bool [is\\_swappable\\_with\\_v](#)
- [error\\_code](#) [make\\_error\\_code](#) (errc) noexcept
- [error\\_condition](#) [make\\_error\\_condition](#) (errc) noexcept
- const nothrow\_t [nothrow](#)
- decltype(nullptr) typedef [nullptr\\_t](#)
- \_GLIBCXX17\_INLINE constexpr [piecewise\\_construct\\_t](#) [piecewise\\_construct](#)
- \_GLIBCXX17\_INLINE constexpr [try\\_to\\_lock\\_t](#) [try\\_to\\_lock](#)

## Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- [istream](#) [cin](#)
- [ostream](#) [cout](#)
- [ostream](#) [cerr](#)
- [ostream](#) [clog](#)
- [wistream](#) [wcin](#)
- [wostream](#) [wcout](#)
- [wostream](#) [wcerr](#)
- [wostream](#) [wclog](#)
- \_\_thread void \* [\\_\\_once\\_callable](#)
- \_\_thread void(\* [\\_\\_once\\_call](#) )()
- template<typename \_Lock >  
[unique\\_lock](#)< \_Lock > [\\_\\_try\\_to\\_lock](#) ( \_Lock &\_\_l)
- template<typename \_Lock1, typename \_Lock2, typename... \_Lock3>  
int [try\\_lock](#) ( \_Lock1 &\_\_l1, \_Lock2 &\_\_l2, \_Lock3 &... \_\_l3)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void [lock](#) ( \_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)
- void [\\_\\_once\\_proxy](#) (void)
- template<typename \_Callable, typename... \_Args>  
void [call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)
- using [\\_\\_shared\\_timed\\_mutex\\_base](#) = [\\_\\_shared\\_mutex\\_cv](#)
- template<typename \_Mutex >  
void [swap](#) ([shared\\_lock](#)< \_Mutex > &\_\_x, [shared\\_lock](#)< \_Mutex > &\_\_y) noexcept

#### 4.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

#### 4.11.2 Typedef Documentation

##### 4.11.2.1 \_\_ptr\_rebind

```
template<typename _Ptr , typename _Tp >  
using std::__ptr_rebind = typedef typename pointer_traits<_Ptr>::template rebind<_Tp>
```

Convenience alias for rebinding pointers.

Definition at line 151 of file ptr\_traits.h.

##### 4.11.2.2 \_\_umap\_traits

```
template<bool _Cache>  
using std::__umap_traits = typedef __detail::_Hashtable_traits<_Cache, false, true>
```

Base types for unordered\_map.

Definition at line 40 of file unordered\_map.h.

##### 4.11.2.3 \_\_ummap\_traits

```
template<bool _Cache>  
using std::__ummap_traits = typedef __detail::_Hashtable_traits<_Cache, false, false>
```

Base types for unordered\_multimap.

Definition at line 57 of file unordered\_map.h.

##### 4.11.2.4 \_\_umset\_traits

```
template<bool _Cache>  
using std::__umset_traits = typedef __detail::_Hashtable_traits<_Cache, true, false>
```

Base types for unordered\_multiset.

Definition at line 55 of file unordered\_set.h.

#### 4.11.2.5 \_\_uset\_traits

```
template<bool _Cache>
using std::__uset_traits = typedef __detail::__Hashtable_traits<_Cache, true, true>
```

Base types for unordered\_set.

Definition at line 40 of file unordered\_set.h.

#### 4.11.2.6 index\_sequence

```
template<size_t... _Idx>
using std::index_sequence = typedef integer_sequence<size_t, _Idx...>
```

Alias template index\_sequence.

Definition at line 336 of file utility.

#### 4.11.2.7 index\_sequence\_for

```
template<typename... _Types>
using std::index_sequence_for = typedef make_index_sequence<sizeof...(_Types)>
```

Alias template index\_sequence\_for.

Definition at line 344 of file utility.

#### 4.11.2.8 make\_index\_sequence

```
template<size_t _Num>
using std::make_index_sequence = typedef make_integer_sequence<size_t, _Num>
```

Alias template make\_index\_sequence.

Definition at line 340 of file utility.

#### 4.11.2.9 make\_integer\_sequence

```
template<typename _Tp, _Tp _Num>
using std::make_integer_sequence = typedef integer_sequence<_Tp, __integer_pack(_Num)...>
```

Alias template make\_integer\_sequence.

Definition at line 329 of file utility.

#### 4.11.2.10 new\_handler

```
typedef void(* std::new_handler) ()
```

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 97 of file `new`.

#### 4.11.2.11 streamoff

```
typedef long long std::streamoff
```

Type used by `fpos`, `char_traits<char>`, and `char_traits<wchar_t>`.

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was typedef `long`.

Definition at line 94 of file `postypes.h`.

#### 4.11.2.12 streampos

```
typedef fpos<mbstate_t> std::streampos
```

File position for char streams.

Definition at line 228 of file `postypes.h`.

#### 4.11.2.13 streamsize

```
typedef ptrdiff_t std::streamsize
```

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file `postypes.h`.

#### 4.11.2.14 terminate\_handler

```
typedef void(* std::terminate_handler) ()
```

If you write a replacement terminate handler, it must be of this type.

Definition at line 61 of file `exception`.

#### 4.11.2.15 u16streampos

```
typedef fpos<mbstate_t> std::u16streampos
```

File position for char16\_t streams.

Definition at line 234 of file postypes.h.

#### 4.11.2.16 u32streampos

```
typedef fpos<mbstate_t> std::u32streampos
```

File position for char32\_t streams.

Definition at line 236 of file postypes.h.

#### 4.11.2.17 unexpected\_handler

```
typedef void(* std::unexpected_handler) ()
```

If you write a replacement unexpected handler, it must be of this type.

Definition at line 64 of file exception.

#### 4.11.2.18 wstreampos

```
typedef fpos<mbstate_t> std::wstreampos
```

File position for wchar\_t streams.

Definition at line 230 of file postypes.h.

### 4.11.3 Enumeration Type Documentation

#### 4.11.3.1 anonymous enum

```
anonymous enum
```

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html) This controls some aspect of the sort routines.

Definition at line 1875 of file stl\_algo.h.

#### 4.11.3.2 float\_denorm\_style

```
enum std::float_denorm_style
```

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the std↵::numeric\_limits class.

**Enumerator**

denorm_indeterminate	Indeterminate at compile time whether denormalized values are allowed.
denorm_absent	The type does not allow denormalized values.
denorm_present	The type allows denormalized values.

Definition at line 182 of file limits.

**4.11.3.3 float\_round\_style**

```
enum std::float_round_style
```

Describes the rounding style for floating-point types.

This is used in the std::numeric\_limits class.

**Enumerator**

round_toward_zero	Intermediate.
round_to_nearest	To zero.
round_toward_infinity	To the nearest representable value.
round_toward_neg_infinity	To infinity.

Definition at line 167 of file limits.

**4.11.3.4 io\_errc**

```
enum std::io_errc [strong]
```

I/O error code.

Definition at line 203 of file ios\_base.h.

**4.11.4 Function Documentation****4.11.4.1 \_\_allocate\_guarded()**

```
template<typename _Alloc >  
__allocated_ptr<_Alloc> std::__allocate_guarded (  
    _Alloc & __a )
```

Allocate space for a single object using \_\_a.

Definition at line 95 of file allocated\_ptr.h.

References std::allocator\_traits< \_Alloc >::allocate().



#### 4.11.4.2 `__final_insertion_sort()`

```
template<typename _RandomAccessIterator , typename _Compare >
void std::__final_insertion_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp )
```

This is a helper function for the sort routine.

Definition at line 1880 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

#### 4.11.4.3 `__find_if()` [1/2]

```
template<typename _InputIterator , typename _Predicate >
_InputIterator std::__find_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred,
    input_iterator_tag ) [inline]
```

This is an overload used by find algos for the Input Iterator case.

Definition at line 101 of file `stl_algo.h`.

Referenced by `__find_if_not()`, and `__search_n_aux()`.

#### 4.11.4.4 `__find_if()` [2/2]

```
template<typename _RandomAccessIterator , typename _Predicate >
_RandomAccessIterator std::__find_if (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Predicate __pred,
    random_access_iterator_tag )
```

This is an overload used by find algos for the RAI case.

Definition at line 112 of file `stl_algo.h`.

#### 4.11.4.5 \_\_find\_if\_not()

```
template<typename _InputIterator , typename _Predicate >
_InputIterator std::__find_if_not (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Provided for `stable_partition` to use.

Definition at line 168 of file `stl_algo.h`.

References `__find_if()`.

#### 4.11.4.6 \_\_find\_if\_not\_n()

```
template<typename _InputIterator , typename _Predicate , typename _Distance >
_InputIterator std::__find_if_not_n (
    _InputIterator __first,
    _Distance & __len,
    _Predicate __pred )
```

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Definition at line 181 of file `stl_algo.h`.

#### 4.11.4.7 \_\_gcd()

```
template<typename _EuclideanRingElement >
_EuclideanRingElement std::__gcd (
    _EuclideanRingElement __m,
    _EuclideanRingElement __n )
```

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1232 of file `stl_algo.h`.

#### 4.11.4.8 \_\_gen\_two\_uniform\_ints()

```
template<typename _IntType , typename _UniformRandomBitGenerator >
pair<_IntType, _IntType> std::__gen_two_uniform_ints (
    _IntType __b0,
    _IntType __b1,
    _UniformRandomBitGenerator && __g )
```

Generate two uniformly distributed integers using a single distribution invocation.

**Parameters**

<code>__b0</code>	The upper bound for the first integer.
<code>__b1</code>	The upper bound for the second integer.
<code>__g</code>	A UniformRandomBitGenerator.

**Returns**

A pair (i, j) with i and j uniformly distributed over [0, \_\_b0) and [0, \_\_b1), respectively.

Requires: `__b0 * __b1 <= __g.max() - __g.min()`.

Using `uniform_int_distribution` with a range that is very small relative to the range of the generator ends up wasting potentially expensively generated randomness, since `uniform_int_distribution` does not store leftover randomness between invocations.

If we know we want two integers in ranges that are sufficiently small, we can compose the ranges, use a single distribution invocation, and significantly reduce the waste.

Definition at line 3769 of file `stl_algo.h`.

References `make_pair()`.

Referenced by `__sample()`.

**4.11.4.9 `__heap_select()`**

```
template<typename _RandomAccessIterator, typename _Compare >
void std::__heap_select (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last,
    _Compare __comp )
```

This is a helper function for the sort routines.

Definition at line 1668 of file `stl_algo.h`.

**4.11.4.10 `__inplace_stable_sort()`**

```
template<typename _RandomAccessIterator, typename _Compare >
void std::__inplace_stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp )
```

This is a helper function for the stable sorting routines.

Definition at line 2761 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

#### 4.11.4.11 \_\_insertion\_sort()

```
template<typename _RandomAccessIterator , typename _Compare >
void std::__insertion_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp )
```

This is a helper function for the sort routine.

Definition at line 1840 of file `stl_algo.h`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

#### 4.11.4.12 \_\_introsort\_loop()

```
template<typename _RandomAccessIterator , typename _Size , typename _Compare >
void std::__introsort_loop (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Size __depth_limit,
    _Compare __comp )
```

This is a helper function for the sort routine.

Definition at line 1940 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

#### 4.11.4.13 \_\_lg()

```
constexpr int std::__lg (
    int __n ) [inline]
```

This is a helper function for the sort routines and for `random.tcc`.

Definition at line 1000 of file `stl_algobase.h`.

Referenced by `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator>()()`, and `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

#### 4.11.4.14 `__merge_adaptive()`

```
template<typename _BidirectionalIterator , typename _Distance , typename _Pointer , typename _Compare >
void std::__merge_adaptive (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    _Distance __len1,
    _Distance __len2,
    _Pointer __buffer,
    _Distance __buffer_size,
    _Compare __comp )
```

This is a helper function for the merge routines.

Definition at line 2415 of file `stl_algo.h`.

#### 4.11.4.15 `__merge_without_buffer()`

```
template<typename _BidirectionalIterator , typename _Distance , typename _Compare >
void std::__merge_without_buffer (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    _Distance __len1,
    _Distance __len2,
    _Compare __comp )
```

This is a helper function for the merge routines.

Definition at line 2476 of file `stl_algo.h`.

References `advance()`, and `iter_swap()`.

Referenced by `__inplace_stable_sort()`.

#### 4.11.4.16 `__move_median_to_first()`

```
template<typename _Iterator , typename _Compare >
void std::__move_median_to_first (
    _Iterator __result,
    _Iterator __a,
    _Iterator __b,
    _Iterator __c,
    _Compare __comp )
```

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__result`.

Definition at line 78 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

**4.11.4.17 \_\_move\_merge()**

```
template<typename _InputIterator , typename _OutputIterator , typename _Compare >
_OutputIterator std::__move_merge (
    _InputIterator __first1,
    _InputIterator __last1,
    _InputIterator __first2,
    _InputIterator __last2,
    _OutputIterator __result,
    _Compare __comp )
```

This is a helper function for the `__merge_sort_loop` routines.

Definition at line 2639 of file `stl_algo.h`.

**4.11.4.18 \_\_move\_merge\_adaptive()**

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
void std::__move_merge_adaptive (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp )
```

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2304 of file `stl_algo.h`.

**4.11.4.19 \_\_move\_merge\_adaptive\_backward()**

```
template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Bidirectional↵
Iterator3 , typename _Compare >
void std::__move_merge_adaptive_backward (
    _BidirectionalIterator1 __first1,
    _BidirectionalIterator1 __last1,
    _BidirectionalIterator2 __first2,
    _BidirectionalIterator2 __last2,
    _BidirectionalIterator3 __result,
    _Compare __comp )
```

This is a helper function for the `__merge_adaptive` routines.




Definition at line 2330 of file `stl_algo.h`.

**4.11.4.20 \_\_once\_proxy()**

```
void std::__once_proxy (
    void )
```

Generic `try_lock`.

**Parameters**

<a href="#"></a> <a href="#"><u>_l1</u></a>	Meets Lockable requirements (try_lock() may throw).
<a href="#"></a> <a href="#"><u>_l2</u></a>	Meets Lockable requirements (try_lock() may throw).
<a href="#"></a> <a href="#"><u>_l3</u></a>	Meets Lockable requirements (try_lock() may throw).

**Returns**

Returns -1 if all try\_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

**Postcondition**

Either all arguments are locked, or none will be.

Sequentially calls try\_lock() on each argument.

**4.11.4.21 \_\_partition()** [1/2]

```
template<typename _ForwardIterator , typename _Predicate >
_FForwardIterator std::__partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    forward_iterator_tag )
```

This is a helper function...

Definition at line 1488 of file stl\_algo.h.

References iter\_swap().

**4.11.4.22 \_\_partition()** [2/2]

```
template<typename _BidirectionalIterator , typename _Predicate >
_BidirectionalIterator std::__partition (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Predicate __pred,
    bidirectional_iterator_tag )
```

This is a helper function...

Definition at line 1513 of file stl\_algo.h.

References iter\_swap().

**4.11.4.23 \_\_reverse()** [1/2]

```
template<typename _BidirectionalIterator >
void std::__reverse (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    bidirectional_iterator_tag )
```

This is an uglified reverse(\_BidirectionalIterator, \_BidirectionalIterator) overloaded for bidirectional iterators.

Definition at line 1132 of file stl\_algo.h.

References iter\_swap().

**4.11.4.24 \_\_reverse()** [2/2]

```
template<typename _RandomAccessIterator >
void std::__reverse (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    random_access_iterator_tag )
```

This is an uglified reverse(\_BidirectionalIterator, \_BidirectionalIterator) overloaded for random access iterators.

Definition at line 1152 of file stl\_algo.h.

References iter\_swap().

**4.11.4.25 \_\_rotate\_adaptive()**

```
template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance
>
_BidirectionalIterator1 std::__rotate_adaptive (
    _BidirectionalIterator1 __first,
    _BidirectionalIterator1 __middle,
    _BidirectionalIterator1 __last,
    _Distance __len1,
    _Distance __len2,
    _BidirectionalIterator2 __buffer,
    _Distance __buffer_size )
```

This is a helper function for the merge routines.

Definition at line 2373 of file stl\_algo.h.



**4.11.4.26 \_\_sample()** [1/2]

```
template<typename _InputIterator , typename _RandomAccessIterator , typename _Size , typename _↵
UniformRandomBitGenerator >
_RandomAccessIterator std::__sample (
    _InputIterator __first,
    _InputIterator __last,
    input_iterator_tag ,
    _RandomAccessIterator __out,
    random_access_iterator_tag ,
    _Size __n,
    _UniformRandomBitGenerator && __g )
```

Reservoir sampling algorithm.

Definition at line 5719 of file stl\_algo.h.

Referenced by \_\_gnu\_parallel::multiseq\_selection().

**4.11.4.27 \_\_sample()** [2/2]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _Cat , typename _Size ,
typename _UniformRandomBitGenerator >
_OutputIterator std::__sample (
    _ForwardIterator __first,
    _ForwardIterator __last,
    forward_iterator_tag ,
    _OutputIterator __out,
    _Cat ,
    _Size __n,
    _UniformRandomBitGenerator && __g )
```

Selection sampling algorithm.

Definition at line 5746 of file stl\_algo.h.

References \_\_gen\_two\_uniform\_ints(), distance(), std::pair< \_T1, \_T2 >::first, min(), and std::pair< \_T1, \_T2 >↵  
::second.

**4.11.4.28 \_\_search\_n\_aux()** [1/2]

```
template<typename _ForwardIterator , typename _Integer , typename _UnaryPredicate >
_FowardIterator std::__search_n_aux (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    _UnaryPredicate __unary_pred,
    std::forward_iterator_tag )
```

This is an helper function for search\_n overloaded for forward iterators.

Definition at line 257 of file stl\_algo.h.

References \_\_find\_if().

4.11.4.29 `__search_n_aux()` [2/2]

```
template<typename _RandomAccessIter , typename _Integer , typename _UnaryPredicate >
_RandomAccessIter std::__search_n_aux (
    _RandomAccessIter __first,
    _RandomAccessIter __last,
    _Integer __count,
    _UnaryPredicate __unary_pred,
    std::random_access_iterator_tag )
```

This is an helper function for `search_n` overloaded for random access iterators.

Definition at line 289 of file `stl_algo.h`.

4.11.4.30 `__stable_partition_adaptive()`

```
template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance
>
_FowardIterator std::__stable_partition_adaptive (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    _Distance __len,
    _Pointer __buffer,
    _Distance __buffer_size )
```

This is a helper function... Requires `__first != __last` and `!__pred(__first)` and `__len == distance(__first, __last)`.

`!__pred(__first)` allows us to guarantee that we don't move-assign an element onto itself.

Definition at line 1549 of file `stl_algo.h`.

4.11.4.31 `__try_to_lock()`

```
template<typename _Lock >
unique_lock<_Lock> std::__try_to_lock (
    _Lock & __l ) [inline]
```

Generic `try_lock`.

## Parameters

<code>__l</code>	Meets Lockable requirements ( <code>try_lock()</code> may throw).
<code>__l2</code>	Meets Lockable requirements ( <code>try_lock()</code> may throw).
<code>__l3</code>	Meets Lockable requirements ( <code>try_lock()</code> may throw).

**Returns**

Returns -1 if all `try_lock()` calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

**Postcondition**

Either all arguments are locked, or none will be.

Sequentially calls `try_lock()` on each argument.

Definition at line 469 of file `mutex`.

**4.11.4.32 `__unguarded_insertion_sort()`**

```
template<typename _RandomAccessIterator, typename _Compare >
void std::__unguarded_insertion_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

This is a helper function for the sort routine.

Definition at line 1863 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

**4.11.4.33 `__unguarded_linear_insert()`**

```
template<typename _RandomAccessIterator, typename _Compare >
void std::__unguarded_linear_insert (
    _RandomAccessIterator __last,
    _Compare __comp )
```

This is a helper function for the sort routine.

Definition at line 1821 of file `stl_algo.h`.

Referenced by `__unguarded_insertion_sort()`.

#### 4.11.4.34 \_\_unguarded\_partition()

```
template<typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::__unguarded_partition (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _RandomAccessIterator __pivot,
    _Compare __comp )
```

This is a helper function...

Definition at line 1896 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

#### 4.11.4.35 \_\_unguarded\_partition\_pivot()

```
template<typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::__unguarded_partition_pivot (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

This is a helper function...

Definition at line 1917 of file `stl_algo.h`.

References `__move_median_to_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

#### 4.11.4.36 \_\_unique\_copy() [1/3]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _BinaryPredicate >
_OutputIterator std::__unique_copy (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred,
    forward_iterator_tag ,
    output_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1049 of file `stl_algo.h`.

**4.11.4.37 \_\_unique\_copy()** [2/3]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
_OutputIterator std::__unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred,
    input_iterator_tag ,
    output_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1078 of file `stl_algo.h`.

**4.11.4.38 \_\_unique\_copy()** [3/3]

```
template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
_FowardIterator std::__unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _ForwardIterator __result,
    _BinaryPredicate __binary_pred,
    input_iterator_tag ,
    forward_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1110 of file `stl_algo.h`.

**4.11.4.39 \_Construct()**

```
template<typename _T1 , typename... _Args>
void std::_Construct (
    _T1 * __p,
    _Args &&... __args ) [inline]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 74 of file `stl_construct.h`.

4.11.4.40 `_Destroy()` [1/3]

```
template<typename _Tp >
void std::_Destroy (
    _Tp * __pointer ) [inline]
```

Destroy the object pointed to by a pointer type.

Definition at line 97 of file `stl_construct.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::~~vector()`.

4.11.4.41 `_Destroy()` [2/3]

```
template<typename _ForwardIterator >
void std::_Destroy (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 127 of file `stl_construct.h`.

4.11.4.42 `_Destroy()` [3/3]

```
template<typename _ForwardIterator , typename _Allocator >
void std::_Destroy (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Allocator & __alloc )
```

Destroy a range of objects using the supplied allocator. For nondefault allocators we do not optimize away invocation of `destroy()` even if `_Tp` has a trivial destructor.

Definition at line 193 of file `stl_construct.h`.

References `__addressof()`.

4.11.4.43 `_Destroy_n()`

```
template<typename _ForwardIterator , typename _Size >
_FForwardIterator std::_Destroy_n (
    _ForwardIterator __first,
    _Size __count ) [inline]
```

Destroy a range of objects. If the `value_type` of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 172 of file `stl_construct.h`.

4.11.4.44 **accumulate()** [1/2]

```
template<typename _InputIterator , typename _Tp >
_Tp std::accumulate (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init ) [inline]
```

Accumulate values in a range.

Accumulates the values in the range [first,last) using operator+(). The initial value is *init*. The values are processed in order.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

**Returns**

The final sum.

Definition at line 120 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

4.11.4.45 **accumulate()** [2/2]

```
template<typename _InputIterator , typename _Tp , typename _BinaryOperation >
_Tp std::accumulate (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init,
    _BinaryOperation __binary_op ) [inline]
```

Accumulate values in a range with operation.

Accumulates the values in the range [first,last) using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.

**Parameters**

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	Function object to accumulate with.

**Returns**

The final sum.

Definition at line 146 of file `stl_numeric.h`.

**4.11.4.46 acos()**

```
template<typename _Tp >
std::complex< _Tp > std::acos (
    const std::complex< _Tp > & __z ) [inline]
```

`acos(__z)` [8.1.2].

Definition at line 1638 of file `complex`.

**4.11.4.47 acosh()**

```
template<typename _Tp >
std::complex< _Tp > std::acosh (
    const std::complex< _Tp > & __z ) [inline]
```

`acosh(__z)` [8.1.5].

Definition at line 1754 of file `complex`.

**4.11.4.48 adjacent\_difference()** [1/2]

```
template<typename _InputIterator , typename _OutputIterator >
_OutputIterator std::adjacent_difference (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[first,last)` using `operator-()` and writes the result to `__result`.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sums.



**Returns**

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 317 of file `stl_numeric.h`.

**4.11.4.49 adjacent\_difference()** [2/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator std::adjacent_difference (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[__first,__last)` using the function object `__binary_op` and writes the result to `__result`.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

**Returns**

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 360 of file `stl_numeric.h`.

**4.11.4.50 advance()**

```
template<typename _InputIterator , typename _Distance >
_GLIBCXX17_CONSTEXPR void std::advance (
    _InputIterator & __i,
    _Distance __n ) [inline]
```

A generalization of pointer arithmetic.

## Parameters

<code>__i</code>	An input iterator.
<code>__n</code>	The <i>delta</i> by which to change <code>__i</code> .

## Returns

Nothing.

This increments `i` by `n`. For bidirectional and random access iterators, `__n` may be negative, in which case `__i` is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 202 of file `std_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__merge_without_buffer()`.

## 4.11.4.51 align()

```
void* std::align (
    size_t __align,
    size_t __size,
    void *& __ptr,
    size_t & __space ) [inline], [noexcept]
```

Fit aligned storage in buffer.

[`ptr.align`]

This function tries to fit `__size` bytes of storage with alignment `__align` into the buffer `__ptr` of size `__space` bytes. If such a buffer fits then `__ptr` is changed to point to the first byte of the aligned storage and `__space` is reduced by the bytes used for alignment.

## Parameters

<code>__align</code>	A fundamental or extended alignment value.
<code>__size</code>	Size of the aligned storage required.
<code>__ptr</code>	Pointer to a buffer of <code>__space</code> bytes.
<code>__space</code>	Size of the buffer pointed to by <code>__ptr</code> .

**Returns**

the updated pointer if the aligned storage fits, otherwise nullptr.

Definition at line 114 of file memory.

**4.11.4.52 arg()**

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::arg (
    _Tp __x ) [inline]
```

Additional overloads [8.1.9].

Definition at line 1852 of file complex.

**4.11.4.53 asin()**

```
template<typename _Tp >
std::complex< _Tp > std::asin (
    const std::complex< _Tp > & __z ) [inline]
```

asin(\_\_z) [8.1.3].

Definition at line 1674 of file complex.

**4.11.4.54 asinh()**

```
template<typename _Tp >
std::complex< _Tp > std::asinh (
    const std::complex< _Tp > & __z ) [inline]
```

asinh(\_\_z) [8.1.6].

Definition at line 1793 of file complex.

**4.11.4.55 atan()**

```
template<typename _Tp >
std::complex< _Tp > std::atan (
    const std::complex< _Tp > & __z ) [inline]
```

atan(\_\_z) [8.1.4].

Definition at line 1718 of file complex.

## 4.11.4.56 atanh()

```
template<typename _Tp >
std::complex< _Tp > std::atanh (
    const std::complex< _Tp > & __z ) [inline]
```

atanh(\_\_z) [8.1.7].

Definition at line 1837 of file complex.

## 4.11.4.57 begin() [1/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::begin (
    _Container & __cont ) -> decltype(__cont.begin()) [inline]
```

Return an iterator pointing to the first element of the container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 48 of file range\_access.h.

## 4.11.4.58 begin() [2/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::begin (
    const _Container & __cont ) -> decltype(__cont.begin()) [inline]
```

Return an iterator pointing to the first element of the const container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 58 of file range\_access.h.

## 4.11.4.59 begin() [3/4]

```
template<typename _Tp , size_t _Nm>
_GLIBCXX14_CONSTEXPR _Tp* std::begin (
    _Tp(&) __arr[_Nm] ) [inline]
```

Return an iterator pointing to the first element of the array.

## Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 87 of file `range_access.h`.

4.11.4.60 `begin()` [4/4]

```
template<class _Tp >
constexpr const _Tp* std::begin (
    initializer_list< _Tp > __ils ) [noexcept]
```

Return an iterator pointing to the first element of the `initializer_list`.

## Parameters

<code>__ils</code>	Initializer list.
--------------------	-------------------

Definition at line 89 of file `initializer_list`.

Referenced by `cbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`, `std::list< __inp, __rebind_inp >::merge()`, `std::list< __inp, __rebind_inp >::remove()`, `std::list< __inp, __rebind_inp >::remove_if()`, `std::forward_list< _Tp, _Alloc >::unique()`, and `std::list< __inp, __rebind_inp >::unique()`.

4.11.4.61 `boolalpha()`

```
ios_base& std::boolalpha (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 880 of file `ios_base.h`.

4.11.4.62 `call_once()`

```
template<typename _Callable , typename... _Args>
void std::call_once (
    once_flag & __once,
    _Callable && __f,
    _Args &&... __args )
```

`call_once`

Definition at line 667 of file `mutex`.

#### 4.11.4.63 cbegin()

```
template<typename _Container >
constexpr auto std::cbegin (
    const _Container & __cont ) -> decltype(std::begin(__cont))    [inline], [noexcept]
```

Return an iterator pointing to the first element of the const container.

##### Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 116 of file range\_access.h.

References [begin\(\)](#).

#### 4.11.4.64 cend()

```
template<typename _Container >
constexpr auto std::cend (
    const _Container & __cont ) -> decltype(std::end(__cont))    [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

##### Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 127 of file range\_access.h.

References [end\(\)](#).

#### 4.11.4.65 const\_pointer\_cast()

```
template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::const_pointer_cast (
    const __shared_ptr< _Tp1, _Lp > & __r )    [inline], [noexcept]
```

`const_pointer_cast`

Definition at line 1559 of file shared\_ptr\_base.h.

#### 4.11.4.66 crbegin()

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::crbegin (
    const _Container & __cont ) -> decltype(std::rbegin(__cont))    [inline]
```

Return a reverse iterator pointing to the last element of the const container.

## Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 218 of file range\_access.h.

References `rbegin()`.

#### 4.11.4.67 `cref()` [1/3]

```
template<typename _Tp >
reference_wrapper<const _Tp> std::cref (
    const _Tp & __t ) [inline], [noexcept]
```

Denotes a const reference should be taken to a variable.

Definition at line 333 of file refwrap.h.

#### 4.11.4.68 `cref()` [2/3]

```
template<typename _Tp >
void std::cref (
    const _Tp && ) [delete]
```

Denotes a reference should be taken to a variable.

#### 4.11.4.69 `cref()` [3/3]

```
template<typename _Tp >
reference_wrapper<const _Tp> std::cref (
    reference_wrapper< _Tp > __t ) [inline], [noexcept]
```

`std::cref` overload to prevent wrapping a `reference_wrapper`

Definition at line 351 of file refwrap.h.

#### 4.11.4.70 `crend()`

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::crend (
    const _Container & __cont ) -> decltype(std::rend(__cont)) [inline]
```

Return a reverse iterator pointing one past the first element of the const container.



**Parameters**

<code>__cont</code>	Container.
---------------------	------------

Definition at line 228 of file `range_access.h`.

References `rend()`.

**4.11.4.71 `dec()`**

```
ios_base& std::dec (  
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 1018 of file `ios_base.h`.

Referenced by operator `>>()`.

**4.11.4.72 `defaultfloat()`**

```
ios_base& std::defaultfloat (  
    ios_base & __base ) [inline]
```

Calls `base.unsetf(ios_base::floatfield)`

Definition at line 1071 of file `ios_base.h`.

**4.11.4.73 `distance()`**

```
template<typename _InputIterator >  
_GLIBCXX17_CONSTEXPR iterator_traits<_InputIterator>::difference_type std::distance (  
    _InputIterator __first,  
    _InputIterator __last ) [inline]
```

A generalization of pointer arithmetic.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 138 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__sample()`, `std::deque<_StateSeqT>::_M_range_initialize()`, `std::sub_match<_Bi_iter>::length()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::match_results<_Bi_iter>::position()`.

**4.11.4.74 dynamic\_pointer\_cast()**

```
template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (
    const __shared_ptr<_Tp1, _Lp > & __r ) [inline], [noexcept]
```

**dynamic\_pointer\_cast**

Definition at line 1572 of file `shared_ptr_base.h`.

**4.11.4.75 end()** [1/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::end (
    _Container & __cont ) -> decltype(__cont.end()) [inline]
```

Return an iterator pointing to one past the last element of the container.

**Parameters**

<code>__cont</code>	Container.
---------------------	------------

Definition at line 68 of file `range_access.h`.

**4.11.4.76 end()** [2/4]

```
template<typename _Container >
```

```
_GLIBCXX17_CONSTEXPR auto std::end (
    const _Container & __cont ) -> decltype(__cont.end())    [inline]
```

Return an iterator pointing to one past the last element of the const container.

#### Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 78 of file range\_access.h.

#### 4.11.4.77 end() [3/4]

```
template<typename _Tp , size_t _Nm>
_GLIBCXX14_CONSTEXPR _Tp* std::end (
    _Tp(&) __arr[_Nm] )    [inline]
```

Return an iterator pointing to one past the last element of the array.

#### Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 97 of file range\_access.h.

#### 4.11.4.78 end() [4/4]

```
template<class _Tp >
constexpr const _Tp* std::end (
    initializer_list< _Tp > __ils )    [noexcept]
```

Return an iterator pointing to one past the last element of the initializer\_list.

#### Parameters

<code>__ils</code>	Initializer list.
--------------------	-------------------

Definition at line 99 of file initializer\_list.

Referenced by cend(), std::vector< sub\_match< \_Bi\_iter >, allocator< sub\_match< \_Bi\_iter > > >::insert(), std::list< \_\_inp, \_\_rebind\_inp >::merge(), std::basic\_filebuf< char\_type, traits\_type >::open(), std::list< \_\_inp, \_\_rebind\_inp >::remove(), std::list< \_\_inp, \_\_rebind\_inp >::remove\_if(), std::forward\_list< \_Tp, \_Alloc >::resize(), std::list< \_\_inp, \_\_rebind\_inp >::resize(), std::forward\_list< \_Tp, \_Alloc >::unique(), and std::list< \_\_inp, \_\_rebind\_inp >::unique().

## 4.11.4.79 endl()

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::endl (
    basic_ostream< _CharT, _Traits > & __os ) [inline]
```

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.↔streambuf.buffering> for more on this subject.

Definition at line 593 of file ostream.

## 4.11.4.80 ends()

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::ends (
    basic_ostream< _CharT, _Traits > & __os ) [inline]
```

Write a null character into the output sequence.

*Null character* is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

Definition at line 605 of file ostream.

## 4.11.4.81 exchange()

```
template<typename _Tp , typename _Up = _Tp>
_Tp std::exchange (
    _Tp & __obj,
    _Up && __new_val ) [inline]
```

Assign `__new_val` to `__obj` and return its previous value.

Definition at line 283 of file utility.

## 4.11.4.82 fabs()

```
template<typename _Tp >
_Tp std::fabs (
    const std::complex< _Tp > & __z ) [inline]
```

`fabs(__z)` [8.1.8].

Definition at line 1846 of file complex.

#### 4.11.4.83 fixed()

```
ios_base& std::fixed (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 1043 of file `ios_base.h`.

#### 4.11.4.84 flush()

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::flush (
    basic_ostream< _CharT, _Traits > & __os ) [inline]
```

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 615 of file `ostream`.

#### 4.11.4.85 get\_money()

```
template<typename _MoneyT >
_Get_money<_MoneyT> std::get_money (
    _MoneyT & __mon,
    bool __intl = false ) [inline]
```

Extended manipulator for extracting money.

##### Parameters

<code>__mon</code>	Either long double or a specialization of <code>basic_string</code> .
<code>__intl</code>	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts `__mon`.

Definition at line 259 of file `iomanip`.

#### 4.11.4.86 get\_new\_handler()

```
new_handler std::get_new_handler ( ) [noexcept]
```

Return the current new handler.

## 4.11.4.87 get\_temporary\_buffer()

```
template<typename _Tp >
pair<_Tp*, ptrdiff_t> std::get_temporary_buffer (
    ptrdiff_t __len ) [noexcept]
```

Allocates a temporary buffer.

## Parameters

<code>__len</code>	The number of objects of type <code>Tp</code> .
--------------------	---

## Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 85 of file `stl_tempbuf.h`.

## 4.11.4.88 get\_terminate()

```
terminate_handler std::get_terminate ( ) [noexcept]
```

Return the current terminate handler.

## 4.11.4.89 get\_time()

```
template<typename _CharT >
_Get_time<_CharT> std::get_time (
    std::tm * __tmb,
    const _CharT * __fmt ) [inline]
```

Extended manipulator for extracting time.

This manipulator uses `time_get::get` to extract time. [ext.manip]

**Parameters**

<code>__tmb</code>	struct to extract the time data to.
<code>__fmt</code>	format string.

Definition at line 413 of file `iomanip`.

**4.11.4.90 `get_unexpected()`**

```
unexpected_handler std::get_unexpected ( ) [noexcept]
```

Return the current unexpected handler.

**4.11.4.91 `getline()`** [1/6]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > & __is,
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    _CharT __delim )
```

Read a line from stream into a string.

**Parameters**

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

Definition at line 627 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` >::`max_size()`.

4.11.4.92 `getline()` [2/6]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_istream<_CharT, _Traits>& std::getline (
    basic_istream< _CharT, _Traits > & __is,
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Read a line from stream into a string.

## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

## Returns

Reference to the input stream.

Stores characters from `is` into `__str` until '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

Definition at line 2676 of file `vstring.h`.

References `getline()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

4.11.4.93 `getline()` [3/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream< _CharT, _Traits > & std::getline (
    basic_istream< _CharT, _Traits > & __is,
    basic_string< _CharT, _Traits, _Alloc > & __str,
    _CharT __delim )
```

Read a line from stream into a string.

## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.



**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If `__delim` is encountered, it is extracted but not stored into `__str`.

Definition at line 1537 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::max_size()`.

Referenced by `getline()`.

**4.11.4.94 `getline()`** [4/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream<_CharT, _Traits>& std::getline (
    basic_istream<_CharT, _Traits > & __is,
    basic_string<_CharT, _Traits, _Alloc > & __str ) [inline]
```

Read a line from stream into a string.

**Parameters**

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until '`'` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If end of line is encountered, it is extracted but not stored into `__str`.

Definition at line 6367 of file `basic_string.h`.

References `getline()`, and `std::basic_ios<_CharT, _Traits>::widen()`.

**4.11.4.95 `getline()`** [5/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream<_CharT, _Traits>& std::getline (
    basic_istream<_CharT, _Traits > && __is,
    basic_string<_CharT, _Traits, _Alloc > & __str,
    _CharT __delim ) [inline]
```

Read a line from an rvalue stream into a string.

Definition at line 6375 of file `basic_string.h`.

References `getline()`.

4.11.4.96 `getline()` [6/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream<_CharT, _Traits>& std::getline (
    basic_istream< _CharT, _Traits > && __is,
    basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]
```

Read a line from an rvalue stream into a string.

Definition at line 6382 of file `basic_string.h`.

References `getline()`.

4.11.4.97 `hex()`

```
ios_base& std::hex (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 1026 of file `ios_base.h`.

Referenced by `std::regex_traits<_CharType>::value()`.

4.11.4.98 `hexfloat()`

```
ios_base& std::hexfloat (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::fixed|ios_basescientific, ios_base::floatfield)`

Definition at line 1063 of file `ios_base.h`.

4.11.4.99 `inner_product()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp >
_Tp std::inner_product (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init ) [inline]
```

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using `operator+()`. The values in the ranges are processed in order.

**Parameters**

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.

**Returns**

The final inner product.

Definition at line 174 of file `stl_numeric.h`.

**4.11.4.100 inner\_product()** [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp , typename _Binary↵
Operation1 , typename _BinaryOperation2 >
_Tp std::inner_product (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init,
    _BinaryOperation1 __binary_op1,
    _BinaryOperation2 __binary_op2 ) [inline]
```

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

**Parameters**

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	Function object to accumulate with.
<code>__binary_op2</code>	Function object to apply to pairs of input values.

**Returns**

The final inner product.

Definition at line 206 of file `stl_numeric.h`.

## 4.11.4.101 internal()

```
ios_base& std::internal (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 993 of file `ios_base.h`.

## 4.11.4.102 iota()

```
template<typename _ForwardIterator , typename _Tp >
void std::iota (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Tp __value )
```

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

## Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

## Returns

Nothing.

Definition at line 82 of file `stl_numeric.h`.

## 4.11.4.103 isalnum()

```
template<typename _CharT >
bool std::isalnum (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

Definition at line 2623 of file `locale_facets.h`.

#### 4.11.4.104 isalpha()

```
template<typename _CharT >
bool std::isalpha (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

Definition at line 2599 of file `locale_facets.h`.

#### 4.11.4.105 isblank()

```
template<typename _CharT >
bool std::isblank (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::blank, __c)`.

Definition at line 2636 of file `locale_facets.h`.

#### 4.11.4.106 iscntrl()

```
template<typename _CharT >
bool std::iscntrl (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

Definition at line 2581 of file `locale_facets.h`.

#### 4.11.4.107 isdigit()

```
template<typename _CharT >
bool std::isdigit (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

Definition at line 2605 of file `locale_facets.h`.

#### 4.11.4.108 isgraph()

```
template<typename _CharT >
bool std::isgraph (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

Definition at line 2629 of file `locale_facets.h`.

#### 4.11.4.109 islower()

```
template<typename _CharT >
bool std::islower (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

Definition at line 2593 of file `locale_facets.h`.

#### 4.11.4.110 isprint()

```
template<typename _CharT >
bool std::isprint (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::print, __c)`.

Definition at line 2575 of file `locale_facets.h`.

#### 4.11.4.111 ispunct()

```
template<typename _CharT >
bool std::ispunct (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

Definition at line 2611 of file `locale_facets.h`.

#### 4.11.4.112 isspace()

```
template<typename _CharT >
bool std::isspace (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::space, __c)`.

Definition at line 2569 of file `locale_facets.h`.

#### 4.11.4.113 isupper()

```
template<typename _CharT >
bool std::isupper (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

Definition at line 2587 of file `locale_facets.h`.

#### 4.11.4.114 isxdigit()

```
template<typename _CharT >
bool std::isxdigit (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

Definition at line 2617 of file `locale_facets.h`.

#### 4.11.4.115 left()

```
ios_base& std::left (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 1001 of file `ios_base.h`.

Referenced by operator<<().

#### 4.11.4.116 lock()

```
template<typename _L1 , typename _L2 , typename... _L3>
void std::lock (
    _L1 & __l1,
    _L2 & __l2,
    _L3 &... __l3 )
```

Generic lock.

## Parameters

$\leftrightarrow$ _l1	Meets Lockable requirements (try_lock() may throw).
$\leftrightarrow$ _l2	Meets Lockable requirements (try_lock() may throw).
$\leftrightarrow$ _l3	Meets Lockable requirements (try_lock() may throw).

## Exceptions

<i>An</i>	exception thrown by an argument's lock() or try_lock() member.
-----------	--

## Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to lock(), try\_lock() and unlock(). If the call exits via an exception any locks that were obtained will be released.

Definition at line 542 of file mutex.

## 4.11.4.117 noboolalpha()

```
ios_base& std::noboolalpha (
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::boolalpha).

Definition at line 888 of file ios\_base.h.

## 4.11.4.118 noshowbase()

```
ios_base& std::noshowbase (
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::showbase).

Definition at line 904 of file ios\_base.h.



**4.11.4.119 noshowpoint()**

```
ios_base& std::noshowpoint (  
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::showpoint).

Definition at line 920 of file ios\_base.h.

**4.11.4.120 noshowpos()**

```
ios_base& std::noshowpos (  
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::showpos).

Definition at line 936 of file ios\_base.h.

**4.11.4.121 noskipws()**

```
ios_base& std::noskipws (  
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::skipws).

Definition at line 952 of file ios\_base.h.

**4.11.4.122 nounitbuf()**

```
ios_base& std::nounitbuf (  
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::unitbuf).

Definition at line 984 of file ios\_base.h.

**4.11.4.123 nouppercase()**

```
ios_base& std::nouppercase (  
    ios_base & __base ) [inline]
```

Calls base.unsetf(ios\_base::uppercase).

Definition at line 968 of file ios\_base.h.

**4.11.4.124 oct()**

```
ios_base& std::oct (
    ios_base & __base ) [inline]
```

Calls base.setf(ios\_base::oct, ios\_base::basefield).

Definition at line 1034 of file ios\_base.h.

Referenced by std::regex\_traits<\_CharType >::value().

**4.11.4.125 operator!=(())** [1/16]

```
template<typename _Tp >
bool std::operator!= (
    const _Fwd_list_iterator< _Tp > & __x,
    const _Fwd_list_const_iterator< _Tp > & __y ) [inline], [noexcept]
```

Forward list iterator inequality comparison.

Definition at line 281 of file forward\_list.h.

**4.11.4.126 operator!=(())** [2/16]

```
template<typename _Tp , typename _Seq >
bool std::operator!= (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y ) [inline]
```

Based on operator==.

Definition at line 329 of file stl\_stack.h.

**4.11.4.127 operator!=(())** [3/16]

```
template<typename _Tp , typename _Seq >
bool std::operator!= (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y ) [inline]
```

Based on operator==.

Definition at line 354 of file stl\_queue.h.

**4.11.4.128 operator!=()** [4/16]

```
template<typename _Res , typename... _Args>
bool std::operator!= (
    const function< _Res(_Args...)> & __f,
    nullptr_t ) [inline], [noexcept]
```

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

**Returns**

false if the wrapper has no target, true otherwise

This function will not throw an exception.

Definition at line 763 of file std\_function.h.

**4.11.4.129 operator!=()** [5/16]

```
template<typename _Res , typename... _Args>
bool std::operator!= (
    nullptr_t ,
    const function< _Res(_Args...)> & __f ) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 769 of file std\_function.h.

**4.11.4.130 operator!=()** [6/16]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator!= (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns !(x == y).

Definition at line 967 of file stl\_multiset.h.

**4.11.4.131 operator!=()** [7/16]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator!= (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns !(x == y).

Definition at line 982 of file stl\_set.h.

**4.11.4.132 operator!=()** [8/16]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator!= (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1122 of file stl\_multimap.h.

**4.11.4.133 operator!=()** [9/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on operator==.

Definition at line 1440 of file forward\_list.h.

**4.11.4.134 operator!=()** [10/16]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator!= (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1458 of file stl\_map.h.

**4.11.4.135 operator!=()** [11/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 1777 of file stl\_vector.h.

**4.11.4.136 operator!=()** [12/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 2028 of file stl\_list.h.

**4.11.4.137 operator!=()** [13/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y ) [inline]
```

Based on operator==.

Definition at line 2303 of file stl\_deque.h.

**4.11.4.138 operator!=()** [14/16]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator!= (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Test difference of two strings.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 6099 of file `basic_string.h`.

**4.11.4.139 operator"!=()" [15/16]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator!= (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test difference of C string and string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 6112 of file `basic_string.h`.

**4.11.4.140 operator"!=()" [16/16]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator!= (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test difference of string and C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 6124 of file basic\_string.h.

#### 4.11.4.141 operator&()

```
template<size_t _Nb>
bitset<_Nb> std::operator& (
    const bitset< _Nb > & __x,
    const bitset< _Nb > & __y ) [inline], [noexcept]
```

Global bitwise operations on bitsets.

##### Parameters

$\leftrightarrow$ __x	A bitset.
$\leftrightarrow$ __y	A bitset of the same size as __x.

##### Returns

A new bitset.

These should be self-explanatory.

Definition at line 1429 of file bitset.

#### 4.11.4.142 operator+() [1/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs )
```

Concatenate two strings.

##### Parameters

__lhs	First string.
__rhs	Last string.

##### Returns

New string with value of \_\_lhs followed by \_\_rhs.

Definition at line 5931 of file basic\_string.h.

**4.11.4.143 operator+()** [2/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs )
```

Concatenate C string and string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with value of `__lhs` followed by `__rhs`.

Definition at line 1157 of file `basic_string.tcc`.

**4.11.4.144 operator+()** [3/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
    _CharT __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs )
```

Concatenate character and string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 1173 of file `basic_string.tcc`.



**4.11.4.145 operator+()** [4/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Concatenate string and C string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 5968 of file `basic_string.h`.

**4.11.4.146 operator+()** [5/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    _CharT __rhs ) [inline]
```

Concatenate string and character.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 5984 of file `basic_string.h`.

**4.11.4.147 operator<()** [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator< (
```

```
const stack< _Tp, _Seq > & __x,  
const stack< _Tp, _Seq > & __y ) [inline]
```

Stack ordering relation.

**Parameters**

$\leftrightarrow$ _x	A stack.
$\leftrightarrow$ _y	A stack of the same type as x.

**Returns**

True iff x is lexicographically less than \_\_y.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 323 of file `stl_stack.h`.

**4.11.4.148 operator<()** [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator< (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y ) [inline]
```

Queue ordering relation.

**Parameters**

$\leftrightarrow$ _x	A queue.
$\leftrightarrow$ _y	A queue of the same type as x.

**Returns**

True iff \_\_x is lexicographically less than \_\_y.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 348 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

## 4.11.4.149 operator&lt;() [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator< (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Multiset ordering relation.

## Parameters

$\leftrightarrow$ __x	A multiset.
$\leftrightarrow$ __y	A multiset of the same type as __x.

## Returns

True iff \_\_x is lexicographically less than \_\_y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <.

See std::lexicographical\_compare() for how the determination is made.

Definition at line 960 of file stl\_multiset.h.

## 4.11.4.150 operator&lt;() [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator< (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Set ordering relation.

## Parameters

$\leftrightarrow$ __x	A set.
$\leftrightarrow$ __y	A set of the same type as x.

## Returns

True iff \_\_x is lexicographically less than \_\_y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 975 of file `stl_set.h`.

#### 4.11.4.151 `operator<()` [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator< (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Multimap ordering relation.

##### Parameters

$\leftrightarrow$ __x	A multimap.
$\leftrightarrow$ __y	A multimap of the same type as __x.

##### Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1115 of file `stl_multimap.h`.

#### 4.11.4.152 `operator<()` [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Forward list ordering relation.

##### Parameters

$\leftrightarrow$ __lx	A forward_list.
$\leftrightarrow$ __ly	A forward_list of the same type as __lx.

**Returns**

True iff `__lx` is lexicographically less than `__ly`.

This is a total ordering relation. It is linear in the number of elements of the forward lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1432 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::cbegin()`, `std::forward_list<_Tp, _Alloc>::cend()`, and `lexicographical_compare()`.

**4.11.4.153 operator<()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator< (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map ordering relation.

**Parameters**

<code>__x</code>	A map.
<code>__y</code>	A map of the same type as <code>x</code> .

**Returns**

True iff `x` is lexicographically less than `y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1451 of file `stl_map.h`.

**4.11.4.154 operator<()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y ) [inline]
```

Vector ordering relation.

**Parameters**

$\_x$	A vector.
$\_y$	A vector of the same type as $\_x$ .

**Returns**

True iff  $\_x$  is lexicographically less than  $\_y$ .

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with  $<$ .

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1770 of file `std_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::end()`, and `lexicographical_compare()`.

**4.11.4.155 operator<() [9/13]**

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
    const list<_Tp, _Alloc> & __x,
    const list<_Tp, _Alloc> & __y ) [inline]
```

List ordering relation.

**Parameters**

$\_x$	A list.
$\_y$	A list of the same type as $\_x$ .

**Returns**

True iff  $\_x$  is lexicographically less than  $\_y$ .

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with  $<$ .

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 2021 of file `std_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, `std::list<_Tp, _Alloc>::end()`, and `lexicographical_compare()`.

## 4.11.4.156 operator&lt;() [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y ) [inline]
```

Deque ordering relation.

## Parameters

$\leftrightarrow$ __x	A deque.
$\leftrightarrow$ __y	A deque of the same type as __x.

## Returns

True iff *x* is lexicographically less than \_\_y.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with <.

See std::lexicographical\_compare() for how the determination is made.

Definition at line 2295 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::begin(), std::deque< \_Tp, \_Alloc >::end(), and lexicographical\_compare().

## 4.11.4.157 operator&lt;() [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator< (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Test if string precedes string.

## Parameters

__lhs	First string.
__rhs	Second string.

## Returns

True if \_\_lhs precedes \_\_rhs. False otherwise.

Definition at line 6137 of file basic\_string.h.



**4.11.4.158** `operator<()` [12/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator< (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string precedes C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 6150 of file `basic_string.h`.

**4.11.4.159** `operator<()` [13/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator< (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string precedes string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 6162 of file `basic_string.h`.

## 4.11.4.160 operator&lt;&lt;() [1/14]

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    _CharT __c ) [inline]
```

Character inserters.

## Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

## Returns

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 500 of file `ostream`.

## 4.11.4.161 operator&lt;&lt;() [2/14]

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    char __c ) [inline]
```

Character inserters.

## Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

## Returns

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 505 of file `ostream`.

#### 4.11.4.162 `operator<<()` [3/14]

```
template<class _Traits >
basic_ostream<char, _Traits>& std::operator<< (
    basic_ostream< char, _Traits > & __out,
    char __c ) [inline]
```

Character inserters.

##### Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

##### Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 511 of file `ostream`.

#### 4.11.4.163 `operator<<()` [4/14]

```
template<class _Traits >
basic_ostream<char, _Traits>& std::operator<< (
    basic_ostream< char, _Traits > & __out,
    signed char __c ) [inline]
```

Character inserters.

##### Parameters

<code>__out</code>	An output stream.
<code>__c</code>	A character.

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 517 of file `ostream`.

**4.11.4.164 operator<<()** [5/14]

```
template<class _Traits >
basic_ostream<char, _Traits>& std::operator<< (
    basic_ostream< char, _Traits > & __out,
    unsigned char __c ) [inline]
```

Character inserters.

**Parameters**

<code>__out</code>	An output stream.
<code>__c</code>	A character.

**Returns**

out

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 522 of file `ostream`.

**4.11.4.165 operator<<()** [6/14]

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    const _CharT * __s ) [inline]
```

String inserters.

**Parameters**

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 542 of file `ostream`.

**4.11.4.166 operator<<() [7/14]**

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::operator<< (
    basic_ostream< _CharT, _Traits > & __out,
    const char * __s )
```

String inserters.

**Parameters**

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 321 of file `ostream.tcc`.

References `std::ios_base::badbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

## 4.11.4.167 operator&lt;&lt;() [8/14]

```
template<class _Traits >
basic_ostream<char, _Traits>& std::operator<< (
    basic_ostream< char, _Traits > & __out,
    const char * __s ) [inline]
```

String inserters.

**Parameters**

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 559 of file ostream.

## 4.11.4.168 operator&lt;&lt;() [9/14]

```
template<class _Traits >
basic_ostream<char, _Traits>& std::operator<< (
    basic_ostream< char, _Traits > & __out,
    const signed char * __s ) [inline]
```

String inserters.

**Parameters**

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2]). `__out.width(0)` is then called.

Definition at line 572 of file `ostream`.

**4.11.4.169 operator<<()** [10/14]

```
template<class _Traits >
basic_ostream<char, _Traits>& std::operator<< (
    basic_ostream< char, _Traits > & __out,
    const unsigned char * __s ) [inline]
```

String inserters.

**Parameters**

<code>__out</code>	An output stream.
<code>__s</code>	A character string.

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2]). `__out.width(0)` is then called.

Definition at line 577 of file `ostream`.

**4.11.4.170 operator<<()** [11/14]

```
template<typename _Ostream , typename _Tp >
enable_if<__and<__not<is_lvalue_reference<_Ostream> >, __is_convertible_to_basic_ostream<↵
_Ostream>, __is_insertable< __rvalue_ostream_type<_Ostream>, const _Tp&> >::value, __rvalue_↵
ostream_type<_Ostream> >::type std::operator<< (
    _Ostream && __os,
    const _Tp & __x ) [inline]
```

Generic inserter for rvalue stream.

## Parameters

<code>__os</code>	An input stream.
<code>__x</code>	A reference to the object being inserted.

## Returns

`os`

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 685 of file ostream.

4.11.4.171 `operator<<()` [12/14]

```
template<class _CharT , class _Traits , size_t _Nb>
std::basic_ostream<_CharT, _Traits>& std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const bitset< _Nb > & __x )
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

Definition at line 1534 of file bitset.

4.11.4.172 `operator<<()` [13/14]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Write string to a stream.

## Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.



**Returns**

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2630 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**4.11.4.173 operator<<()** [14/14]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]
```

Write string to a stream.

**Parameters**

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

**Returns**

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 6327 of file `basic_string.h`.

**4.11.4.174 operator<=()** [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator<= (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y ) [inline]
```

Based on `operator<`.

Definition at line 341 of file `stl_stack.h`.

**4.11.4.175 operator<=()** [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator<= (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y ) [inline]
```

Based on operator<.

Definition at line 366 of file stl\_queue.h.

**4.11.4.176 operator<=()** [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator<= (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns !(y < x)

Definition at line 981 of file stl\_multiset.h.

**4.11.4.177 operator<=()** [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator<= (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns !(y < x)

Definition at line 996 of file stl\_set.h.

**4.11.4.178 operator<=()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator<= (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 1136 of file stl\_multimap.h.

**4.11.4.179 operator<=()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on operator<.

Definition at line 1461 of file forward\_list.h.

**4.11.4.180 operator<=()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator<= (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 1472 of file stl\_map.h.

**4.11.4.181 operator<=()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 1789 of file stl\_vector.h.

**4.11.4.182 operator<=()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 2040 of file stl\_list.h.

**4.11.4.183 operator<=()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 2317 of file `std_deque.h`.

**4.11.4.184 operator<=()** [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator<= (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Test if string doesn't follow string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 6213 of file `basic_string.h`.

**4.11.4.185 operator<=()** [12/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator<= (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string doesn't follow C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 6226 of file `basic_string.h`.

**4.11.4.186 operator<=()** [13/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator<= (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string doesn't follow string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 6238 of file `basic_string.h`.

**4.11.4.187 operator==()** [1/17]

```
template<typename _StateT >
bool std::operator== (
    const fpos< _StateT > & __lhs,
    const fpos< _StateT > & __rhs ) [inline]
```

Test if equivalent to another position.

Definition at line 216 of file `postypes.h`.

**4.11.4.188 operator==()** [2/17]

```
template<typename _Tp >
bool std::operator== (
    const _Fwd_list_iterator< _Tp > & __x,
    const _Fwd_list_const_iterator< _Tp > & __y ) [inline], [noexcept]
```

Forward list iterator equality comparison.

Definition at line 272 of file `forward_list.h`.

## 4.11.4.189 operator==( ) [3/17]

```
template<typename _Tp , typename _Seq >
bool std::operator==(
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y ) [inline]
```

Stack equality comparison.

## Parameters

$\_x$	A stack.
$\_y$	A stack of the same type as $\_x$ .

## Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 305 of file stl\_stack.h.

## 4.11.4.190 operator==( ) [4/17]

```
template<typename _Tp , typename _Seq >
bool std::operator==(
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y ) [inline]
```

Queue equality comparison.

## Parameters

$\_x$	A queue.
$\_y$	A queue of the same type as $\_x$ .

**Returns**

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 330 of file `std_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**4.11.4.191 operator==( ) [5/17]**

```
template<typename _Res , typename... _Args>
bool std::operator==(
    const function< _Res(_Args...)> & __f,
    nullptr_t ) [inline], [noexcept]
```

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

**Returns**

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 745 of file `std_function.h`.

**4.11.4.192 operator==( ) [6/17]**

```
template<typename _Res , typename... _Args>
bool std::operator==(
    nullptr_t ,
    const function< _Res(_Args...)> & __f ) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 751 of file `std_function.h`.

**4.11.4.193 operator==( ) [7/17]**

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator==(
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Multiset equality comparison.

**Parameters**

$\_x$	A multiset.
$\_y$	A multiset of the same type as $\_x$ .

**Returns**

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 943 of file `stl_multiset.h`.

**4.11.4.194 operator==()** [8/17]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator== (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Set equality comparison.

**Parameters**

$\_x$	A set.
$\_y$	A set of the same type as $\_x$ .

**Returns**

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 958 of file `stl_set.h`.



**4.11.4.195 operator==()** [9/17]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator== (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Multimap equality comparison.

**Parameters**

$\_x$	A multimap.
$\_y$	A multimap of the same type as $\_x$ .

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1098 of file `std_multimap.h`.

**4.11.4.196 operator==()** [10/17]

```
template<typename _Tp , typename _Alloc >
bool std::operator== (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly )
```

Forward list equality comparison.

**Parameters**

$\_x$	A forward_list
$\_y$	A forward_list of the same type as $\_x$ .

**Returns**

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 369 of file `forward_list.tcc`.

**4.11.4.197 operator==()** [11/17]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator== (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map equality comparison.

**Parameters**

$\_x$	A map.
$\_y$	A map of the same type as $\_x$ .

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1434 of file `std_map.h`.

**4.11.4.198 operator==()** [12/17]

```
template<typename _Tp, typename _Alloc >
bool std::operator== (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y ) [inline]
```

Vector equality comparison.

**Parameters**

$\_x$	A vector.
$\_y$	A vector of the same type as $\_x$ .

**Returns**

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1753 of file `std_vector.h`.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equal()`, and `std::vector< _Tp, _Alloc >::size()`.

4.11.4.199 `operator==()` [13/17]

```
template<typename _Tp , typename _Alloc >
_GLIBCXX_END_NAMESPACE_CXX11 bool std::operator== (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y ) [inline]
```

List equality comparison.

## Parameters

<code>__x</code>	A list.
<code>__y</code>	A list of the same type as <code>__x</code> .

## Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1987 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, and `std::list< _Tp, _Alloc >::size()`.

4.11.4.200 `operator==()` [14/17]

```
template<typename _Tp , typename _Alloc >
bool std::operator== (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y ) [inline]
```

Deque equality comparison.

## Parameters

<code>__x</code>	A deque.
<code>__y</code>	A deque of the same type as <code>__x</code> .

## Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the dequeues. Dequeues are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 2277 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, `equal()`, and `std::deque<_Tp, _Alloc>::size()`.

#### 4.11.4.201 operator==( [15/17] )

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator==(
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Test equivalence of two strings.

##### Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

##### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 6052 of file `basic_string.h`.

#### 4.11.4.202 operator==( [16/17] )

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator==(
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test equivalence of C string and string.

##### Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 6074 of file `basic_string.h`.

**4.11.4.203 operator==( )** [17/17]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator==(
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test equivalence of string and C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 6086 of file `basic_string.h`.

**4.11.4.204 operator>( )** [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator>(
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y ) [inline]
```

Based on `operator<`.

Definition at line 335 of file `stl_stack.h`.

**4.11.4.205 operator>( )** [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator>(
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y ) [inline]
```

Based on `operator<`.

Definition at line 360 of file `stl_queue.h`.

**4.11.4.206 operator>()** [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator> (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns  $y < x$ .

Definition at line 974 of file `stl_multiset.h`.

**4.11.4.207 operator>()** [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator> (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns  $y < x$ .

Definition at line 989 of file `stl_set.h`.

**4.11.4.208 operator>()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator> (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on `operator<`.

Definition at line 1129 of file `stl_multimap.h`.

**4.11.4.209 operator>()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.

Definition at line 1447 of file `forward_list.h`.



**4.11.4.210 operator>()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator> (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 1465 of file stl\_map.h.

**4.11.4.211 operator>()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 1783 of file stl\_vector.h.

**4.11.4.212 operator>()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 2034 of file stl\_list.h.

**4.11.4.213 operator>()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 2310 of file stl\_deque.h.

**4.11.4.214 operator>()** [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator> (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Test if string follows string.

## Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 6175 of file `basic_string.h`.

4.11.4.215 `operator>()` [12/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator> (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string follows C string.

## Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 6188 of file `basic_string.h`.

4.11.4.216 `operator>()` [13/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator> (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string follows string.

## Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 6200 of file `basic_string.h`.

**4.11.4.217 operator>=()** [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator>= (
    const stack< _Tp, _Seq > & __x,
    const stack< _Tp, _Seq > & __y ) [inline]
```

Based on `operator<`.

Definition at line 347 of file `stl_stack.h`.

**4.11.4.218 operator>=()** [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator>= (
    const queue< _Tp, _Seq > & __x,
    const queue< _Tp, _Seq > & __y ) [inline]
```

Based on `operator<`.

Definition at line 372 of file `stl_queue.h`.

**4.11.4.219 operator>=()** [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator>= (
    const multiset< _Key, _Compare, _Alloc > & __x,
    const multiset< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns `!(x < y)`

Definition at line 988 of file `stl_multiset.h`.

**4.11.4.220 operator>=()** [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator>= (
    const set< _Key, _Compare, _Alloc > & __x,
    const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Returns  $!(x < y)$

Definition at line 1003 of file `stl_set.h`.

**4.11.4.221 operator>=()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator>= (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    const multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on `operator<`.

Definition at line 1143 of file `stl_multimap.h`.

**4.11.4.222 operator>=()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.

Definition at line 1454 of file `forward_list.h`.

**4.11.4.223 operator>=()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator>= (
    const map< _Key, _Tp, _Compare, _Alloc > & __x,
    const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Based on `operator<`.

Definition at line 1479 of file `stl_map.h`.

**4.11.4.224 operator>=()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
    const vector< _Tp, _Alloc > & __x,
    const vector< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 1795 of file stl\_vector.h.

**4.11.4.225 operator>=()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
    const list< _Tp, _Alloc > & __x,
    const list< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 2046 of file stl\_list.h.

**4.11.4.226 operator>=()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
    const deque< _Tp, _Alloc > & __x,
    const deque< _Tp, _Alloc > & __y ) [inline]
```

Based on operator<.

Definition at line 2324 of file stl\_deque.h.

**4.11.4.227 operator>=()** [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator>= (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Test if string doesn't precede string.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 6251 of file `basic_string.h`.

**4.11.4.228 operator>=()** [12/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator>= (
    const basic_string< _CharT, _Traits, _Alloc > & __lhs,
    const _CharT * __rhs ) [inline]
```

Test if string doesn't precede C string.

**Parameters**

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 6264 of file `basic_string.h`.

**4.11.4.229 operator>=()** [13/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator>= (
    const _CharT * __lhs,
    const basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string doesn't precede string.

**Parameters**

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 6276 of file basic\_string.h.

#### 4.11.4.230 operator>>() [1/11]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __in,
    _CharT & __c )
```

Character extractors.

##### Parameters

$\leftrightarrow$ __in	An input stream.
$\leftrightarrow$ __c	A character reference.

##### Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic\_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in \_\_c. Otherwise, sets failbit in the input stream.

Definition at line 931 of file istream.tcc.

References std::ios\_base::goodbit.

#### 4.11.4.231 operator>>() [2/11]

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
    basic_istream< char, _Traits > & __in,
    unsigned char & __c ) [inline]
```

Character extractors.

##### Parameters

$\leftrightarrow$ __in	An input stream.
$\leftrightarrow$ __c	A character reference.

**Returns**

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

Definition at line 756 of file `istream`.

**4.11.4.232 operator>>() [3/11]**

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
    basic_istream< char, _Traits > & __in,
    signed char & __c ) [inline]
```

Character extractors.

**Parameters**

<code>__in</code>	An input stream.
<code>__c</code>	A character reference.

**Returns**

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

Definition at line 761 of file `istream`.

**4.11.4.233 operator>>() [4/11]**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __in,
    _CharT * __s )
```

Character string extractors.



**Parameters**

<code>__in</code>	An input stream.
<code>__s</code>	A pointer to a character array.

**Returns**`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 963 of file `istream.tcc`.

References `std::ios_base::goodbit`.

**4.11.4.234 operator>>() [5/11]**

```
template<>
basic_istream<char>& std::operator>> (
    basic_istream< char > & __in,
    char * __s )
```

Character string extractors.

## Parameters

$\leftrightarrow$ __in	An input stream.
$\leftrightarrow$ __s	A pointer to a character array.

## Returns

\_\_in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

## 4.11.4.235 operator&gt;&gt;() [6/11]

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
    basic_istream< char, _Traits > & __in,
    unsigned char * __s ) [inline]
```

Character string extractors.

## Parameters

$\leftrightarrow$ __in	An input stream.
$\leftrightarrow$ __s	A pointer to a character array.

**Returns**`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 803 of file `istream`.

**4.11.4.236 operator>>() [7/11]**

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
    basic_istream< char, _Traits > & __in,
    signed char * __s ) [inline]
```

Character string extractors.

**Parameters**

<code>__in</code>	An input stream.
<code>__s</code>	A pointer to a character array.

**Returns**`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 808 of file `istream`.

**4.11.4.237 operator>>() [8/11]**

```
template<typename _Istream , typename _Tp >
enable_if<__and<__not<is_lvalue_reference<_Istream> >, __is_convertible_to_basic_istream<_Istream>, __is_extractable<__rvalue_istream_type<_Istream>, _Tp&&> >::value, __rvalue_istream_type<_Istream> >::type std::operator>> (
    _Istream && __is,
    _Tp && __x ) [inline]
```

Generic extractor for rvalue stream.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A reference to the extraction target.

**Returns**

is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 980 of file istream.

**4.11.4.238 operator>>() [9/11]**

```
template<class _CharT , class _Traits , size_t _Nb>
std::basic_istream<_CharT, _Traits>& std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    bitset< _Nb > & __x )
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

Definition at line 1466 of file bitset.

**4.11.4.239 operator>>() [10/11]**

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str )
```

Read stream into a string.

**Parameters**

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 552 of file vstring.tcc.

## 4.11.4.240 operator&gt;&gt;() [11/11]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream< _CharT, _Traits > & std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    basic_string< _CharT, _Traits, _Alloc > & __str )
```

Read stream into a string.

## Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

## Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 1465 of file `basic_string.tcc`.

## 4.11.4.241 operator^()

```
template<size_t _Nb>
bitset<_Nb> std::operator^ (
    const bitset< _Nb > & __x,
    const bitset< _Nb > & __y ) [inline], [noexcept]
```

Global bitwise operations on bitsets.

## Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

## Returns

A new bitset.

These should be self-explanatory.

Definition at line 1447 of file `bitset`.

4.11.4.242 `operator" |()`

```
template<size_t _Nb>
bitset<_Nb> std::operator| (
    const bitset< _Nb > & __x,
    const bitset< _Nb > & __y ) [inline], [noexcept]
```

Global bitwise operations on bitsets.

## Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

## Returns

A new bitset.

These should be self-explanatory.

Definition at line 1438 of file `bitset`.

4.11.4.243 `partial_sum()` [1/2]

```
template<typename _InputIterator , typename _OutputIterator >
_OutputIterator std::partial_sum (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result )
```

Return list of partial sums.

Accumulates the values in the range `[first,last)` using the `+` operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

## Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.

**Returns**

Iterator pointing just beyond the values written to `__result`.

Definition at line 237 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs_pu()`, and `__gnu_parallel::__sequential_random_shuffle()`.

**4.11.4.244 partial\_sum()** [2/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator std::partial_sum (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op )
```

Return list of partial sums.

Accumulates the values in the range `[first,last)` using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

**Parameters**

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

**Returns**

Iterator pointing just beyond the values written to `__result`.

Definition at line 278 of file `stl_numeric.h`.

**4.11.4.245 put\_money()**

```
template<typename _MoneyT >
_Put_money<_MoneyT> std::put_money (
    const _MoneyT & __mon,
    bool __intl = false ) [inline]
```

Extended manipulator for inserting money.



**Parameters**

<code>__mon</code>	Either long double or a specialization of <code>basic_string</code> .
<code>__intl</code>	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts `__mon`.

Definition at line 306 of file `iomanip`.

**4.11.4.246 put\_time()**

```
template<typename _CharT >
_Put_time<_CharT> std::put_time (
    const std::tm * __tmb,
    const _CharT * __fmt ) [inline]
```

Extended manipulator for formatting time.

This manipulator uses `time_put::put` to format time. [ext.manip]

**Parameters**

<code>__tmb</code>	struct <code>tm</code> time data to format.
<code>__fmt</code>	format string.

Definition at line 358 of file `iomanip`.

**4.11.4.247 quoted()**

```
template<typename _CharT >
auto std::quoted (
    const _CharT * __string,
    _CharT __delim = _CharT('\"'),
    _CharT __escape = _CharT('\\') ) [inline]
```

Manipulator for quoted strings.

**Parameters**

<code>__string</code>	String to quote.
<code>__delim</code>	Character to quote string with.
<code>__escape</code>	Escape character to escape itself or quote character.

Definition at line 461 of file `iomanip`.

**4.11.4.248 rbegin()** [1/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::rbegin (
    _Container & __cont ) -> decltype(__cont.rbegin())    [inline]
```

Return a reverse iterator pointing to the last element of the container.

**Parameters**

<code>__cont</code>	Container.
---------------------	------------

Definition at line 138 of file `range_access.h`.

Referenced by `crbegin()`.

**4.11.4.249 rbegin()** [2/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::rbegin (
    const _Container & __cont ) -> decltype(__cont.rbegin())    [inline]
```

Return a reverse iterator pointing to the last element of the const container.

**Parameters**

<code>__cont</code>	Container.
---------------------	------------

Definition at line 148 of file `range_access.h`.

**4.11.4.250 rbegin()** [3/4]

```
template<typename _Tp , size_t _Nm>
_GLIBCXX17_CONSTEXPR reverse\_iterator<_Tp*> std::rbegin (
    _Tp(&) __arr[_Nm] )    [inline]
```

Return a reverse iterator pointing to the last element of the array.

**Parameters**

<code>__arr</code>	Array.
--------------------	--------

Definition at line 178 of file range\_access.h.

#### 4.11.4.251 `rbegin()` [4/4]

```
template<typename _Tp >
_GLIBCXX17_CONSTEXPR reverse_iterator<const _Tp*> std::rbegin (
    initializer_list< _Tp > __il ) [inline]
```

Return a reverse iterator pointing to the last element of the `initializer_list`.

##### Parameters

$\leftrightarrow$ <code>__il</code>	<code>initializer_list</code> .
--	---------------------------------

Definition at line 198 of file range\_access.h.

#### 4.11.4.252 `ref()` [1/3]

```
template<typename _Tp >
reference_wrapper<_Tp> std::ref (
    _Tp & __t ) [inline], [noexcept]
```

Denotes a reference should be taken to a variable.

Definition at line 327 of file refwrap.h.

#### 4.11.4.253 `ref()` [2/3]

```
template<typename _Tp >
void std::ref (
    const _Tp && ) [delete]
```

Denotes a reference should be taken to a variable.

#### 4.11.4.254 `ref()` [3/3]

```
template<typename _Tp >
reference_wrapper<_Tp> std::ref (
    reference_wrapper< _Tp > __t ) [inline], [noexcept]
```

`std::ref` overload to prevent wrapping a `reference_wrapper`

Definition at line 345 of file refwrap.h.

**4.11.4.255** `rend()` [1/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::rend (
    _Container & __cont ) -> decltype(__cont.rend())    [inline]
```

Return a reverse iterator pointing one past the first element of the container.

**Parameters**

<code>__cont</code>	Container.
---------------------	------------

Definition at line 158 of file `range_access.h`.

Referenced by `crend()`.

**4.11.4.256** `rend()` [2/4]

```
template<typename _Container >
_GLIBCXX17_CONSTEXPR auto std::rend (
    const _Container & __cont ) -> decltype(__cont.rend())    [inline]
```

Return a reverse iterator pointing one past the first element of the const container.

**Parameters**

<code>__cont</code>	Container.
---------------------	------------

Definition at line 168 of file `range_access.h`.

**4.11.4.257** `rend()` [3/4]

```
template<typename _Tp , size_t _Nm>
_GLIBCXX17_CONSTEXPR reverse\_iterator<_Tp*> std::rend (
    _Tp(&) __arr[_Nm] )    [inline]
```

Return a reverse iterator pointing one past the first element of the array.

**Parameters**

<code>__arr</code>	Array.
--------------------	--------

Definition at line 188 of file `range_access.h`.

4.11.4.258 `rend()` [4/4]

```
template<typename _Tp >
_GLIBCXX17_CONSTEXPR reverse_iterator<const _Tp*> std::rend (
    initializer_list< _Tp > __il ) [inline]
```

Return a reverse iterator pointing one past the first element of the `initializer_list`.

## Parameters

<code>__il</code>	<code>initializer_list</code> .
-------------------	---------------------------------

Definition at line 208 of file `range_access.h`.

4.11.4.259 `replace_copy()`

```
template<typename _InputIterator , typename _OutputIterator , typename _Tp >
_OutputIterator std::replace_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    const _Tp & __old_value,
    const _Tp & __new_value ) [inline]
```

Copy a sequence, replacing each element of one value with another value.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

## Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range [`__first`,`__last`) to the output range [`__result`,`__result+(__last-__first)`) replacing elements equal to `__old_value` with `__new_value`.

Definition at line 3136 of file `stl_algo.h`.

## 4.11.4.260 resetiosflags()

```
_Resetiosflags std::resetiosflags (
    ios_base::fmtflags __mask ) [inline]
```

Manipulator for `setf`.

## Parameters

<code>__mask</code>	A format flags mask.
---------------------	----------------------

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0, __mask)`.

Definition at line 66 of file `iosmanip`.

## 4.11.4.261 return\_temporary\_buffer()

```
template<typename _Tp >
void std::return_temporary_buffer (
    _Tp * __p ) [inline]
```

The companion to `get_temporary_buffer()`.

## Parameters

<code>__p</code>	A buffer previously allocated by <code>get_temporary_buffer</code> .
------------------	--

## Returns

None.

Frees the memory pointed to by `__p`.

Definition at line 112 of file `stl_tempbuf.h`.

## 4.11.4.262 right()

```
ios_base& std::right (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 1009 of file `ios_base.h`.

**4.11.4.263 scientific()**

```
ios_base& std::scientific (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 1051 of file `ios_base.h`.

Referenced by operator<<().

**4.11.4.264 set\_new\_handler()**

```
new_handler std::set_new_handler (
    new_handler ) throw ( )
```

Takes a replacement handler as the argument, returns the previous handler.

**4.11.4.265 set\_terminate()**

```
terminate_handler std::set_terminate (
    terminate_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

**4.11.4.266 set\_unexpected()**

```
unexpected_handler std::set_unexpected (
    unexpected_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

**4.11.4.267 setbase()**

```
_Setbase std::setbase (
    int __base ) [inline]
```

Manipulator for `setf`.

**Parameters**

<code>__base</code>	A numeric base.
---------------------	-----------------

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when `base` is 8, 10, or 16, accordingly, and to 0 if `__base` is any other value.

Definition at line 127 of file `iomanip`.

#### 4.11.4.268 `setfill()`

```
template<typename _CharT >
_Setfill<_CharT> std::setfill (
    _CharT __c ) [inline]
```

Manipulator for `fill`.

##### Parameters

<code>__c</code>	The new fill character.
------------------	-------------------------

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

Definition at line 165 of file `iomanip`.

#### 4.11.4.269 `setiosflags()`

```
_Setiosflags std::setiosflags (
    ios_base::fmtflags __mask ) [inline]
```

Manipulator for `setf`.

##### Parameters

<code>__mask</code>	A format flags mask.
---------------------	----------------------

Sent to a stream object, this manipulator sets the format flags to `__mask`.

Definition at line 96 of file `iomanip`.

#### 4.11.4.270 `setprecision()`

```
_Setprecision std::setprecision (
    int __n ) [inline]
```

Manipulator for `precision`.



**Parameters**

$\_n$	The new precision.
-------	--------------------

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

Definition at line 195 of file `iomanip`.

**4.11.4.271 setw()**

```
_Setw std::setw (
    int __n ) [inline]
```

Manipulator for width.

**Parameters**

$\_n$	The new width.
-------	----------------

Sent to a stream object, this manipulator calls `width(__n)` for that object.

Definition at line 225 of file `iomanip`.

**4.11.4.272 showbase()**

```
ios_base& std::showbase (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::showbase)`.

Definition at line 896 of file `ios_base.h`.

**4.11.4.273 showpoint()**

```
ios_base& std::showpoint (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::showpoint)`.

Definition at line 912 of file `ios_base.h`.

**4.11.4.274 showpos()**

```
ios_base& std::showpos (
    ios_base & __base ) [inline]
```

Calls base.setf(ios\_base::showpos).

Definition at line 928 of file ios\_base.h.

**4.11.4.275 skipws()**

```
ios_base& std::skipws (
    ios_base & __base ) [inline]
```

Calls base.setf(ios\_base::skipws).

Definition at line 944 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>(), and operator>>().

**4.11.4.276 static\_pointer\_cast()**

```
template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::static_pointer_cast (
    const __shared_ptr< _Tp1, _Lp > & __r ) [inline], [noexcept]
```

static\_pointer\_cast

Definition at line 1546 of file shared\_ptr\_base.h.

**4.11.4.277 swap()** [1/18]

```
template<typename _Res , typename... _Args>
void std::swap (
    function< _Res(_Args...)> & __x,
    function< _Res(_Args...)> & __y ) [inline], [noexcept]
```

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 784 of file std\_function.h.

**4.11.4.278 swap()** [2/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
    basic_stringbuf< _CharT, _Traits, _Allocator > & __x,
    basic_stringbuf< _CharT, _Traits, _Allocator > & __y ) [inline]
```

Swap specialization for stringbufs.

Definition at line 797 of file sstream.

**4.11.4.279 swap()** [3/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
    basic_istream< _CharT, _Traits, _Allocator > & __x,
    basic_istream< _CharT, _Traits, _Allocator > & __y ) [inline]
```

Swap specialization for istreams.

Definition at line 804 of file sstream.

**4.11.4.280 swap()** [4/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
    basic_ostringstream< _CharT, _Traits, _Allocator > & __x,
    basic_ostringstream< _CharT, _Traits, _Allocator > & __y ) [inline]
```

Swap specialization for ostringstreams.

Definition at line 811 of file sstream.

**4.11.4.281 swap()** [5/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
    basic_stringstream< _CharT, _Traits, _Allocator > & __x,
    basic_stringstream< _CharT, _Traits, _Allocator > & __y ) [inline]
```

Swap specialization for stringstream.

Definition at line 818 of file sstream.

**4.11.4.282 swap()** [6/18]

```
template<typename _Key , typename _Compare , typename _Alloc >
void std::swap (
    multiset< _Key, _Compare, _Alloc > & __x,
    multiset< _Key, _Compare, _Alloc > & __y ) [inline], [noexcept]
```

See `std::multiset::swap()`.

Definition at line 995 of file `stl_multiset.h`.

**4.11.4.283 swap()** [7/18]

```
template<typename _Key , typename _Compare , typename _Alloc >
void std::swap (
    set< _Key, _Compare, _Alloc > & __x,
    set< _Key, _Compare, _Alloc > & __y ) [inline], [noexcept]
```

See `std::set::swap()`.

Definition at line 1010 of file `stl_set.h`.

**4.11.4.284 swap()** [8/18]

```
template<class _CharT , class _Traits >
void std::swap (
    basic_filebuf< _CharT, _Traits > & __x,
    basic_filebuf< _CharT, _Traits > & __y ) [inline]
```

Swap specialization for filebufs.

Definition at line 1140 of file `fstream`.

**4.11.4.285 swap()** [9/18]

```
template<class _CharT , class _Traits >
void std::swap (
    basic_ifstream< _CharT, _Traits > & __x,
    basic_ifstream< _CharT, _Traits > & __y ) [inline]
```

Swap specialization for ifstreams.

Definition at line 1147 of file `fstream`.

**4.11.4.286 swap()** [10/18]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
void std::swap (
    multimap< _Key, _Tp, _Compare, _Alloc > & __x,
    multimap< _Key, _Tp, _Compare, _Alloc > & __y ) [inline], [noexcept]
```

See `std::multimap::swap()`.

Definition at line 1150 of file `stl_multimap.h`.

**4.11.4.287 swap()** [11/18]

```
template<class _CharT , class _Traits >
void std::swap (
    basic_ofstream< _CharT, _Traits > & __x,
    basic_ofstream< _CharT, _Traits > & __y ) [inline]
```

Swap specialization for ofstreams.

Definition at line 1154 of file `fstream`.

**4.11.4.288 swap()** [12/18]

```
template<class _CharT , class _Traits >
void std::swap (
    basic_fstream< _CharT, _Traits > & __x,
    basic_fstream< _CharT, _Traits > & __y ) [inline]
```

Swap specialization for fstreams.

Definition at line 1161 of file `fstream`.

**4.11.4.289 swap()** [13/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
    forward_list< _Tp, _Alloc > & __lx,
    forward_list< _Tp, _Alloc > & __ly ) [inline], [noexcept]
```

See `std::forward_list::swap()`.

Definition at line 1468 of file `forward_list.h`.

**4.11.4.290 swap()** [14/18]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
void std::swap (
    map< _Key, _Tp, _Compare, _Alloc > & __x,
    map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline], [noexcept]
```

See `std::map::swap()`.

Definition at line 1486 of file `stl_map.h`.

**4.11.4.291 swap()** [15/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
    vector< _Tp, _Alloc > & __x,
    vector< _Tp, _Alloc > & __y ) [inline], [noexcept]
```

See `std::vector::swap()`.

Definition at line 1801 of file `stl_vector.h`.

**4.11.4.292 swap()** [16/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
    list< _Tp, _Alloc > & __x,
    list< _Tp, _Alloc > & __y ) [inline], [noexcept]
```

See `std::list::swap()`.

Definition at line 2052 of file `stl_list.h`.

**4.11.4.293 swap()** [17/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
    deque< _Tp, _Alloc > & __x,
    deque< _Tp, _Alloc > & __y ) [inline], [noexcept]
```

See `std::deque::swap()`.

Definition at line 2331 of file `stl_deque.h`.

**4.11.4.294 swap()** [18/18]

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::swap (
    basic_string< _CharT, _Traits, _Alloc > & __lhs,
    basic_string< _CharT, _Traits, _Alloc > & __rhs ) [inline], [noexcept]
```

Swap contents of two strings.

**Parameters**

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 6289 of file `basic_string.h`.

**4.11.4.295 `terminate()`**

```
void std::terminate ( ) [noexcept]
```

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

**4.11.4.296 `tolower()`**

```
template<typename _CharT >
_CharT std::tolower (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype::tolower(__c)`.

Definition at line 2649 of file `locale_facets.h`.

**4.11.4.297 `toupper()`**

```
template<typename _CharT >
_CharT std::toupper (
    _CharT __c,
    const locale & __loc ) [inline]
```

Convenience interface to `ctype::toupper(__c)`.

Definition at line 2643 of file `locale_facets.h`.

**4.11.4.298 `try_lock()`**

```
template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
int std::try_lock (
    _Lock1 & __l1,
    _Lock2 & __l2,
    _Lock3 &... __l3 )
```

Generic `try_lock`.

**Parameters**

<code>_↔ _l1</code>	Meets Lockable requirements (try_lock() may throw).
<code>_↔ _l2</code>	Meets Lockable requirements (try_lock() may throw).
<code>_↔ _l3</code>	Meets Lockable requirements (try_lock() may throw).

**Returns**

Returns -1 if all try\_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

**Postcondition**

Either all arguments are locked, or none will be.

Sequentially calls try\_lock() on each argument.

Definition at line 521 of file mutex.

**4.11.4.299 uncaught\_exception()**

```
_GLIBCXX17_DEPRECATED bool std::uncaught_exception ( ) [noexcept]
```

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering unexpected() due to the throw; or after entering terminate() for any reason other than an explicit call to terminate(). [Note: This includes stack unwinding [15.2]. end note]'

2: 'When uncaught\_exception() is true, throwing an exception can result in a call of terminate() (15.5.1).'

**4.11.4.300 uncaught\_exceptions()**

```
int std::uncaught_exceptions ( ) [noexcept]
```

The number of uncaught exceptions.

**4.11.4.301 unexpected()**

```
void std::unexpected ( )
```

The runtime will call this function if an exception is thrown which violates the function's exception specification.

**4.11.4.302 uninitialized\_copy()**

```
template<typename _InputIterator , typename _ForwardIterator >
_FowardIterator std::uninitialized_copy (
    _InputIterator __first,
    _InputIterator __last,
    _ForwardIterator __result ) [inline]
```

Copies the range [first,last) into result.



**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

**Returns**

`__result + (__first - __last)`

Like `copy()`, but does not require an initialized output range.

Definition at line 115 of file `stl_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

**4.11.4.303 uninitialized\_copy\_n()**

```
template<typename _InputIterator , typename _Size , typename _ForwardIterator >
_FowardIterator std::uninitialized_copy_n (
    _InputIterator __first,
    _Size __n,
    _ForwardIterator __result ) [inline]
```

Copies the range `[first,first+n)` into `result`.

**Parameters**

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

**Returns**

`__result + __n`

Like `copy_n()`, but does not require an initialized output range.

Definition at line 812 of file `stl_uninitialized.h`.

References `__iterator_category()`.

## 4.11.4.304 uninitialized\_fill()

```
template<typename _ForwardIterator , typename _Tp >
void std::uninitialized_fill (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __x ) [inline]
```

Copies the value x into the range [first,last).

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	The source value.

## Returns

Nothing.

Like fill(), but does not require an initialized output range.

Definition at line 181 of file stl\_uninitialized.h.

## 4.11.4.305 uninitialized\_fill\_n()

```
template<typename _ForwardIterator , typename _Size , typename _Tp >
_FowardIterator std::uninitialized_fill_n (
    _ForwardIterator __first,
    _Size __n,
    const _Tp & __x ) [inline]
```

Copies the value x into the range [first,first+n).

## Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of copies to make.
<code>__x</code>	The source value.

## Returns

Nothing.

Like fill\_n(), but does not require an initialized output range.

Definition at line 244 of file stl\_uninitialized.h.

#### 4.11.4.306 unitbuf()

```
ios_base& std::unitbuf (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 976 of file `ios_base.h`.

#### 4.11.4.307 uppercase()

```
ios_base& std::uppercase (
    ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::uppercase)`.

Definition at line 960 of file `ios_base.h`.

#### 4.11.4.308 ws()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::ws (
    basic_istream< _CharT, _Traits > & __is )
```

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;
std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling `operator>>` on `cin` and your object. Note that the same effect can be achieved by creating a `std::basic_istream::sentry` inside your definition of `operator>>`.

Definition at line 1024 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

### 4.11.5 Variable Documentation

#### 4.11.5.1 \_\_iointit

```
ios_base::Init std::__iointit [static]
```

Linked to standard error (buffered)




Definition at line 74 of file `iostream`.

#### 4.11.5.2 \_\_once\_call

```
__thread void(* std::__once_call) ()
```

Generic `try_lock`.

**Parameters**

 _l1	Meets Lockable requirements (try_lock() may throw).
 _l2	Meets Lockable requirements (try_lock() may throw).
 _l3	Meets Lockable requirements (try_lock() may throw).

**Returns**

Returns -1 if all try\_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

**Postcondition**

Either all arguments are locked, or none will be.




Sequentially calls try\_lock() on each argument.

**4.11.5.3 \_\_once\_callable**

```
__thread void* std::__once_callable
```

Generic try\_lock.

**Parameters**

 _l1	Meets Lockable requirements (try_lock() may throw).
 _l2	Meets Lockable requirements (try_lock() may throw).
 _l3	Meets Lockable requirements (try_lock() may throw).

**Returns**

Returns -1 if all try\_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

**Postcondition**

Either all arguments are locked, or none will be.

Sequentially calls try\_lock() on each argument.

#### 4.11.5.4 cerr

```
ostream std::cerr
```

Linked to standard output.

#### 4.11.5.5 cin

```
istream std::cin
```

Linked to standard input.

#### 4.11.5.6 clog

```
ostream std::clog
```

Linked to standard error (unbuffered)

#### 4.11.5.7 cout

```
ostream std::cout
```

Linked to standard input.

#### 4.11.5.8 is\_nothrow\_swappable\_v

```
template<typename _Tp >  
_GLIBCXX17_INLINE constexpr bool std::is_nothrow_swappable_v
```

is\_nothrow\_swappable\_v

Definition at line 2477 of file type\_traits.

#### 4.11.5.9 is\_nothrow\_swappable\_with\_v

```
template<typename _Tp , typename _Up >  
_GLIBCXX17_INLINE constexpr bool std::is_nothrow_swappable_with_v
```

is\_nothrow\_swappable\_with\_v

Definition at line 2561 of file type\_traits.

#### 4.11.5.10 is\_swappable\_v

```
template<typename _Tp >  
_GLIBCXX17_INLINE constexpr bool std::is_swappable_v
```

is\_swappable\_v

Definition at line 2472 of file type\_traits.

#### 4.11.5.11 is\_swappable\_with\_v

```
template<typename _Tp , typename _Up >  
_GLIBCXX17_INLINE constexpr bool std::is_swappable_with_v
```

is\_swappable\_with\_v

Definition at line 2556 of file type\_traits.

#### 4.11.5.12 wcerr

```
wostream std::wcerr
```

Linked to standard output.

#### 4.11.5.13 wcin

```
wistream std::wcin
```

Linked to standard error (buffered)

#### 4.11.5.14 wlog

```
wostream std::wlog
```

Linked to standard error (unbuffered)

#### 4.11.5.15 wcout

```
wostream std::wcout
```

Linked to standard input.

## 4.12 std::\_\_debug Namespace Reference

### Classes

- class [bitset](#)
- class [deque](#)
- class [forward\\_list](#)
- class [list](#)
- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)
- class [vector](#)

### Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp && get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>  
bool operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _←  
_Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >  
&__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_set< _Value,  
_Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,  
_Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,  
_Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`



- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`enable_if< !::__array_traits< _Tp, _Nm >::is_swappable::value >::type swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, std::size_t _Nm>`  
`void swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__↵`  
`y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__↵`  
`y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator`  
`> &__rhs) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash,`  
`_Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`  
`noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.↵`  
`swap(__ly)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash,`  
`_Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp,`  
`_Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

#### 4.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.

Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__↵`  
`_glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling  
with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the  
standard library algorithms.

#### 4.12.2 Function Documentation

##### 4.12.2.1 operator<=()

```
template<typename _Tp, typename _Alloc >
bool std::__debug::operator<= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.

Definition at line 820 of file `debug/forward_list`.

## 4.12.2.2 operator&gt;()

```
template<typename _Tp , typename _Alloc >
bool std::__debug::operator> (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on operator<.

Definition at line 806 of file debug/forward\_list.

## 4.12.2.3 operator&gt;=()

```
template<typename _Tp , typename _Alloc >
bool std::__debug::operator>= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on operator<.

Definition at line 813 of file debug/forward\_list.

## 4.12.2.4 swap()

```
template<typename _Tp , typename _Alloc >
void std::__debug::swap (
    forward_list< _Tp, _Alloc > & __lx,
    forward_list< _Tp, _Alloc > & __ly ) [inline], [noexcept]
```

See std::forward\_list::swap().

Definition at line 827 of file debug/forward\_list.

## 4.13 std::\_\_detail Namespace Reference

## Classes

- struct [\\_BracketMatcher](#)
- class [\\_Compiler](#)
- struct [\\_Default\\_ranged\\_hash](#)
- struct [\\_Equal\\_helper](#)
- struct [\\_Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, false >](#)
- struct [\\_Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, true >](#)
- struct [\\_Equality](#)
- struct [\\_Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >](#)
- struct [\\_Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >](#)

- struct [\\_Equality\\_base](#)
- class [\\_Executor](#)
- struct [\\_Hash\\_code\\_base](#)
- struct [\\_Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, false >](#)
- struct [\\_Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, true >](#)
- struct [\\_Hash\\_code\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, false >](#)
- struct [\\_Hash\\_node](#)
- struct [\\_Hash\\_node<\\_Value, false >](#)
- struct [\\_Hash\\_node<\\_Value, true >](#)
- struct [\\_Hash\\_node\\_base](#)
- struct [\\_Hash\\_node\\_value\\_base](#)
- struct [\\_Hashtable\\_alloc](#)
- struct [\\_Hashtable\\_base](#)
- struct [\\_Hashtable\\_ebo\\_helper](#)
- struct [\\_Hashtable\\_ebo\\_helper<\\_Nm, \\_Tp, false >](#)
- struct [\\_Hashtable\\_ebo\\_helper<\\_Nm, \\_Tp, true >](#)
- struct [\\_Hashtable\\_traits](#)
- struct [\\_Insert](#)
- struct [\\_Insert<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >](#)
- struct [\\_Insert<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >](#)
- struct [\\_Insert\\_base](#)
- struct [\\_List\\_node\\_base](#)
- struct [\\_List\\_node\\_header](#)
- struct [\\_Local\\_const\\_iterator](#)
- struct [\\_Local\\_iterator](#)
- struct [\\_Local\\_iterator\\_base](#)
- struct [\\_Local\\_iterator\\_base<\\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, true >](#)
- struct [\\_Map\\_base](#)
- struct [\\_Map\\_base<\\_Key, \\_Pair, \\_Alloc, \\_Select1st, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >](#)
- struct [\\_Map\\_base<\\_Key, \\_Pair, \\_Alloc, \\_Select1st, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >](#)
- struct [\\_Mask\\_range\\_hashing](#)
- struct [\\_Mod\\_range\\_hashing](#)
- struct [\\_Node\\_const\\_iterator](#)
- struct [\\_Node\\_iterator](#)
- struct [\\_Node\\_iterator\\_base](#)
- struct [\\_Power2\\_rehash\\_policy](#)
- struct [\\_Prime\\_rehash\\_policy](#)
- struct [\\_Quoted\\_string](#)
- struct [\\_Rehash\\_base](#)
- struct [\\_Rehash\\_base<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, std::false\\_type >](#)
- struct [\\_Rehash\\_base<\\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, std::true\\_type >](#)
- class [\\_Scanner](#)
- class [\\_StateSeq](#)

## Typedefs

- template<typename \_Iter, typename \_TraitsT >  
using **\_\_disable\_if\_contiguous\_normal\_iter** = typename [enable\\_if](#)< !\_\_is\_contiguous\_normal\_iter< \_Iter >↔  
::value, [std::shared\\_ptr](#)< const \_NFA< \_TraitsT > > >::type
- template<typename \_Iter, typename \_TraitsT >  
using **\_\_enable\_if\_contiguous\_normal\_iter** = typename [enable\\_if](#)< \_\_is\_contiguous\_normal\_iter< \_Iter >↔  
::value, [std::shared\\_ptr](#)< const \_NFA< \_TraitsT > > >::type
- template<typename \_Policy >  
using **\_\_has\_load\_factor** = typename \_Policy::\_\_has\_load\_factor
- template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_H1, typename \_H2, typename \_Hash >  
using **\_\_hash\_code\_for\_local\_iter** = \_Hash\_code\_storage< [\\_Hash\\_code\\_base](#)< \_Key, \_Value, \_ExtractKey,  
\_H1, \_H2, \_Hash, false > >
- template<typename \_CharT >  
using **\_\_Matcher** = std::function< bool(\_CharT)>
- typedef long **\_\_StateIdT**

## Enumerations

- enum [\\_Opcode](#) : int {  
    **\_S\_opcode\_unknown**, **\_S\_opcode\_alternative**, **\_S\_opcode\_repeat**, **\_S\_opcode\_backref**,  
    **\_S\_opcode\_line\_begin\_assertion**, **\_S\_opcode\_line\_end\_assertion**, **\_S\_opcode\_word\_boundary**, **\_S\_**↔  
    **opcode\_subexpr\_lookahead**,  
    **\_S\_opcode\_subexpr\_begin**, **\_S\_opcode\_subexpr\_end**, **\_S\_opcode\_dummy**, **\_S\_opcode\_match**,  
    **\_S\_opcode\_accept** }  
• enum **\_\_RegexExecutorPolicy** : int { **\_S\_auto**, **\_S\_alterate** }

## Functions

- template<typename \_Up, typename \_Tp >  
constexpr \_Up **\_\_absu** (\_Tp \_\_val)
- template<typename \_Up >  
void **\_\_absu** (bool)=delete
- [\\_GLIBCXX14\\_CONSTEXPR](#) std::size\_t **\_\_clp2** (std::size\_t \_\_n) noexcept
- template<typename \_TraitsT, typename \_FwdIter >  
\_\_enable\_if\_contiguous\_normal\_iter< \_FwdIter, \_TraitsT > **\_\_compile\_nfa** (\_FwdIter \_\_first, \_FwdIter \_\_last,  
const typename \_TraitsT::locale\_type &\_\_loc, [regex\\_constants::syntax\\_option\\_type](#) \_\_flags)
- template<typename \_TraitsT, typename \_FwdIter >  
\_\_disable\_if\_contiguous\_normal\_iter< \_FwdIter, \_TraitsT > **\_\_compile\_nfa** (\_FwdIter \_\_first, \_FwdIter \_\_last,  
const typename \_TraitsT::locale\_type &\_\_loc, [regex\\_constants::syntax\\_option\\_type](#) \_\_flags)
- template<class \_Iterator >  
std::iterator\_traits< \_Iterator >::difference\_type **\_\_distance\_fw** (\_Iterator \_\_first, \_Iterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<class \_Iterator >  
std::iterator\_traits< \_Iterator >::difference\_type **\_\_distance\_fw** (\_Iterator \_\_first, \_Iterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<class \_Iterator >  
std::iterator\_traits< \_Iterator >::difference\_type **\_\_distance\_fw** (\_Iterator \_\_first, \_Iterator \_\_last)
- template<typename \_ValT, typename \_CharT, typename \_Traits >  
[basic\\_istream](#)< \_CharT, \_Traits > & **\_\_extract\_params** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [vector](#)< \_ValT  
> &\_\_vals, size\_t \_\_n)
- template<typename \_Tp >  
constexpr \_Tp **\_\_gcd** (\_Tp \_\_m, \_Tp \_\_n)

- `template<typename _Tp >`  
`constexpr _Tp __lcm (_Tp __m, _Tp __n)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator __normalize (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__factor)`
- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT, _RegexExecutorPolicy __policy, bool __match_↵`  
`mode>`  
`bool __regex_algo_impl (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > &__m, const basic_regex<`  
`_CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags)`
- `template<typename _Tp >`  
`bool __Power_of_2 (_Tp __x)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base<`  
`_Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x,`  
`const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`_Quoted_string< const _CharT *, _CharT > &__str)`
- `template<typename _CharT, typename _Traits, typename _String >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`_Quoted_string< _String, _CharT > &__str)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool operator== (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base<`  
`_Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x,`  
`const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, const`  
`_Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)`

## Variables

- `static const _StateIdT _S_invalid_state_id`

### 4.13.1 Detailed Description

Implementation details not part of the namespace `std` interface.

### 4.13.2 Function Documentation

## 4.13.2.1 operator&lt;&lt;() [1/2]

```
template<typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& std::__detail::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const _Quoted_string< const _CharT *, _CharT > & __str )
```

Insertter for quoted strings.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS DR 2344 quoted()'s interaction with padding is unclear

Definition at line 93 of file quoted\_string.h.

References std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >::str().

## 4.13.2.2 operator&lt;&lt;() [2/2]

```
template<typename _CharT , typename _Traits , typename _String >
std::basic_ostream<_CharT, _Traits>& std::__detail::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const _Quoted_string< _String, _CharT > & __str )
```

Insertter for quoted strings.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS DR 2344 quoted()'s interaction with padding is unclear

Definition at line 117 of file quoted\_string.h.

References std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >::str().

## 4.13.2.3 operator&gt;&gt;()

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_istream<_CharT, _Traits>& std::__detail::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > & __str )
```

Extractor for delimited strings. The left and right delimiters can be different.

Definition at line 139 of file quoted\_string.h.

References std::basic\_ios< \_CharT, \_Traits >::clear(), std::ios\_base::flags(), std::basic\_ios< \_CharT, \_Traits >::good(), std::ios\_base::setf(), std::skipws(), and std::basic\_istream< \_CharT, \_Traits >::unset().

## 4.14 std::\_\_parallel Namespace Reference

## Classes

- struct [\\_CRandNumber](#)



## Functions

- `template<typename _Iter, typename _Tp, typename _Tag >`  
`_Tp __accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Tp __accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`  
`_Tp __accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`  
`_Tp __accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAlter, typename _Tp, typename _BinaryOper >`  
`_Tp __accumulate_switch (_RAlter, _RAlter, _Tp, _BinaryOper, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism  
__parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _RAlter, typename _Tp, typename _BinaryOperation >`  
`_Tp __accumulate_switch (_RAlter __begin, _RAlter __end, _Tp __init, _BinaryOperation __binary_op,  
random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary↵  
Operation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary↵  
Operation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism ↵  
__parallelism_tag)`
- `template<typename _Filter, typename _IterTag >`  
`_Filter __adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_Filter __adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAlter, typename _BiPredicate >`  
`_RAlter __adjacent_find_switch (_RAlter, _RAlter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _RAlter >`  
`_RAlter __adjacent_find_switch (_RAlter __begin, _RAlter __end, random\_access\_iterator\_tag)`
- `template<typename _Filterator, typename _IteratorTag >`  
`_Filterator __adjacent_find_switch (_Filterator __begin, _Filterator __end, _IteratorTag)`
- `template<typename _Filterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_Filterator __adjacent_find_switch (_Filterator __begin, _Filterator __end, _BinaryPredicate __pred, _Iterator↵  
Tag)`
- `template<typename _RAlter, typename _BinaryPredicate >`  
`_RAlter __adjacent_find_switch (_RAlter __begin, _RAlter __end, _BinaryPredicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAlter, typename _Predicate >`  
`iterator_traits< _RAlter >::difference_type __count_if_switch (_RAlter __begin, _RAlter __end, _Predicate ↵  
__pred, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred,  
_IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`

- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type __count_switch ( _RAIter __begin, _RAIter __end, const _Tp &__value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type __count_switch ( _Iter __begin, _Iter __end, const _Tp &__value, __IteratorTag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool __equal_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool __equal_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch ( _Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_RAIter __find_first_of_switch ( _RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random\_access\_iterator\_tag, __IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Iter __find_first_of_switch ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_RAIter __find_first_of_switch ( _RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random\_access\_iterator\_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Iter __find_first_of_switch ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter __find_if_switch ( _Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`_Iter __find_if_switch ( _Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter __find_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter __find_switch ( _Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter __find_switch ( _RAIter __begin, _RAIter __end, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter __find_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`  
`_Function __for_each_switch ( _Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function __for_each_switch ( _RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function __for_each_switch ( _Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter __generate_n_switch ( _OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`  
`_OutputIterator __generate_n_switch ( _OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`

- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag,`  
`gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIter, typename _Generator, typename _IterTag >`  
`void __generate_switch (_FIter, _FIter, _Generator, _IterTag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`  
`void __generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`  
`void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag,`  
`gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`  
`_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2,`  
`random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism=gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename`  
`_Tag1, typename _Tag2 >`  
`_Tp __inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp __inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _Binary↵`  
`Function1 __binary_op1, _BinaryFunction2 __binary_op2, random\_access\_iterator\_tag, random\_access\_iterator\_tag,`  
`gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename`  
`_IteratorTag1, typename _IteratorTag2 >`  
`_Tp __inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 ↵`  
`__binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool __lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _Compare, typename _IterTag >`  
`_FIter __max_element_switch (_FIter, _FIter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag,`  
`gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2,`  
`typename _IterTag3 >`  
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random\_access\_iterator\_tag,`  
`random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename ↵`  
`IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag,`  
`random\_access\_iterator\_tag)`

- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`  
`_Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator __partition_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void __replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`  
`void __replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void __replace_switch (_FIterator __begin, _FIterator __end, const _Tp & __old_value, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`

- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_Filter __search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, random\_access\_iterator\_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_Filterator __search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`

- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OutputIterator __set_symmetric_difference_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAlter , typename _Predicate >`  
`_Output_RAlter __set_symmetric_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >`  
`_OIter __set_symmetric_difference_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, __↵  
IterTag2, _IterTag3)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OutputIterator __set_union_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __↵  
OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAlter , typename _Predicate >`  
`_Output_RAlter __set_union_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __↵  
end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >`  
`_OIter __set_union_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter , typename _OIter , typename _UnaryOperation , typename _IterTag1 , typename _IterTag2 >`  
`_OIter __transform1_switch ( _Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter , typename _RAOIter , typename _UnaryOperation >`  
`_RAOIter __transform1_switch ( _RAIter, _RAIter, _RAOIter, _UnaryOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation >`  
`_RAIter2 __transform1_switch ( _RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __↵  
unary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __↵  
parallelism_tag)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >`  
`_RAIter2 __transform1_switch ( _RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __↵  
unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BiOperation >`  
`_RAIter3 __transform2_switch ( _RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1 , typename _Iter2 , typename _OIter , typename _BiOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >`  
`_OIter __transform2_switch ( _Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BinaryOperation >`  
`_RAIter3 __transform2_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __↵  
result, _BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _BinaryOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >`  
`_OutputIterator __transform2_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __↵  
result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter , typename _OutputIterator , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >`  
`_OutputIterator __unique_copy_switch ( _Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`

- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`  
`RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`  
`_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`



- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation`  
`__binary_op)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred)`



- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __↵  
parallelism_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __↵  
parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filterator >`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Compare >`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator >`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism  
\_\_parallelism\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter merge (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __↵  
result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __↵  
result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __↵  
result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __↵  
result)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism  
\_\_parallelism\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`

- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter >`  
`void nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum ( _Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum ( _Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum ( _Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum ( _Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary↵`  
`__op)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filterator, typename _Predicate >`  
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Predicate >`  
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`
- `template<typename _RAlter >`  
`void random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filterator, typename _Tp >`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Tp >`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp >`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, ↵`  
`_BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, ↵`  
`_BinaryPredicate __pred)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate`  
`__binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↵`  
`Iterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↵`  
`Iterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`





- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`



- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`

#### 4.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

## 4.15 std::\_\_profile Namespace Reference

## Classes

- class [bitset](#)
- class [deque](#)
- class [forward\\_list](#)
- class [list](#)
- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)

## Functions

- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>  
bool __are_equal (const _UnorderedCont &__uc, const \_\_detail::\_\_Hash\_node< _Value, _Cache_hash_code >  
* __lhs, const \_\_detail::\_\_Hash\_node< _Value, _Cache_hash_code > * __rhs)`
- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>  
std::size_t __get_bucket_index (const _UnorderedCont &__uc, const \_\_detail::\_\_Hash\_node< _Value, _Cache_↵  
_hash_code > * __node)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _↵  
IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >  
bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator,  
_Sequence > &__rhs) noexcept`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered\_set< _Key, _↵  
Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_map<  
_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_multiset<  
_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_multimap<  
_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >`  
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _`  
`Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,`  
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,`  
`_Allocator > &__rhs)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > operator+ (typename __iterator_tracker< _Iterator, _Sequence >::`  
`difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type operator- (const __iterator_tracker< _IteratorL,`  
`_Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type operator- (const __iterator_tracker< _Iterator, _`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _`  
`IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator,`  
`_Sequence > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >`  
`&__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _`  
`Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,`  
`_Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,`  
`_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`bitset< _Nb > &__x)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.<- swap(__ly)))`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, unordered_set< _Key, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

#### 4.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

#### 4.15.2 Function Documentation

##### 4.15.2.1 `operator<=()`

```
template<typename _Tp , typename _Alloc >
bool std::__profile::operator<= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.

Definition at line 202 of file `profile/forward_list`.

##### 4.15.2.2 `operator>()`

```
template<typename _Tp , typename _Alloc >
bool std::__profile::operator> (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.

Definition at line 188 of file `profile/forward_list`.

##### 4.15.2.3 `operator>=()`

```
template<typename _Tp , typename _Alloc >
bool std::__profile::operator>= (
    const forward_list< _Tp, _Alloc > & __lx,
    const forward_list< _Tp, _Alloc > & __ly ) [inline]
```

Based on `operator<`.

Definition at line 195 of file `profile/forward_list`.

## 4.15.2.4 swap()

```
template<typename _Tp , typename _Alloc >
void std::__profile::swap (
    forward_list< _Tp, _Alloc > & __lx,
    forward_list< _Tp, _Alloc > & __ly ) [inline], [noexcept]
```

See `std::forward_list::swap()`.

Definition at line 209 of file `profile/forward_list`.

## 4.16 std::chrono Namespace Reference

## Classes

- struct [duration](#)
- struct [duration\\_values](#)
- struct [time\\_point](#)
- struct [treat\\_as\\_floating\\_point](#)

## Typedefs

- template<typename \_Rep1 , typename \_Rep2 , typename \_CRep = typename common\_type<\_Rep1, \_Rep2>::type>  
using **\_\_common\_rep\_t** = typename [enable\\_if](#)< [is\\_convertible](#)< const \_Rep2 &, \_CRep >::value, \_CRep >::type
- template<typename \_Tp >  
using **\_\_disable\_if\_is\_duration** = typename [enable\\_if](#)<!\_\_is\_duration< \_Tp >::value, \_Tp >::type
- template<typename \_Tp >  
using **\_\_enable\_if\_is\_duration** = typename [enable\\_if](#)< \_\_is\_duration< \_Tp >::value, \_Tp >::type
- typedef [duration](#)< int64\_t, ratio< 3600 > > [hours](#)
- typedef [duration](#)< int64\_t, micro > [microseconds](#)
- typedef [duration](#)< int64\_t, milli > [milliseconds](#)
- typedef [duration](#)< int64\_t, ratio< 60 > > [minutes](#)
- typedef [duration](#)< int64\_t, nano > [nanoseconds](#)
- typedef [duration](#)< int64\_t > [seconds](#)

## Functions

- template<typename \_ToDur , typename \_Rep , typename \_Period >  
constexpr **\_\_enable\_if\_is\_duration**< \_ToDur > [duration\\_cast](#) (const [duration](#)< \_Rep, \_Period > & \_\_d)
- template<typename \_Rep1 , typename \_Period1 , typename \_Rep2 , typename \_Period2 >  
constexpr bool **operator!=** (const [duration](#)< \_Rep1, \_Period1 > & \_\_lhs, const [duration](#)< \_Rep2, \_Period2 > & \_\_rhs)
- template<typename \_Clock , typename \_Dur1 , typename \_Dur2 >  
constexpr bool **operator!=** (const [time\\_point](#)< \_Clock, \_Dur1 > & \_\_lhs, const [time\\_point](#)< \_Clock, \_Dur2 > & \_\_rhs)
- template<typename \_Rep1 , typename \_Period , typename \_Rep2 >  
constexpr [duration](#)< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 > >, \_Period > **operator%** (const [duration](#)< \_Rep1, \_Period > & \_\_d, const \_Rep2 & \_\_s)



- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator%`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _`  
`Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Rep2, typename _Period >`  
`constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > operator* (const _Rep1 &__s, const`  
`duration< _Rep2, _Period > &__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > op-`  
`erator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > op-`  
`erator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator-`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr common_type< _Dur1, _Dur2 >::type operator- (const time_point< _Clock, _Dur1 > &__lhs, const`  
`time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 >, _Period > operator/`  
`(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _Period1 > &__lhs, const`  
`duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator== (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`

- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator>= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 >`  
`&__rhs)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`  
`constexpr enable_if< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > >::type time_point_cast`  
`(const time_point< _Clock, _Dur > &__t)`

#### 4.16.1 Detailed Description

ISO C++ 2011 entities sub-namespace for time and date.

#### 4.16.2 Typedef Documentation

##### 4.16.2.1 hours

```
typedef duration<int64_t, ratio<3600> > std::chrono::hours
```

hours

Definition at line 612 of file chrono.

##### 4.16.2.2 microseconds

```
typedef duration<int64_t, micro> std::chrono::microseconds
```

microseconds

Definition at line 600 of file chrono.

##### 4.16.2.3 milliseconds

```
typedef duration<int64_t, milli> std::chrono::milliseconds
```

milliseconds

Definition at line 603 of file chrono.

#### 4.16.2.4 minutes

```
typedef duration<int64_t, ratio< 60> > std::chrono::minutes
```

minutes

Definition at line 609 of file chrono.

#### 4.16.2.5 nanoseconds

```
typedef duration<int64_t, nano> std::chrono::nanoseconds
```

nanoseconds

Definition at line 597 of file chrono.

#### 4.16.2.6 seconds

```
typedef duration<int64_t> std::chrono::seconds
```

seconds

Definition at line 606 of file chrono.

### 4.16.3 Function Documentation

#### 4.16.3.1 duration\_cast()

```
template<typename _ToDur , typename _Rep , typename _Period >  
constexpr __enable_if_is_duration<_ToDur> std::chrono::duration_cast (   
    const duration< _Rep, _Period > & __d )
```

duration\_cast

Definition at line 193 of file chrono.

## 4.16.3.2 time\_point\_cast()

```
template<typename _ToDur , typename _Clock , typename _Dur >
constexpr enable_if<__is_duration<_ToDur>::value, time_point<_Clock, _ToDur> >::type std::chrono<
::time_point_cast (
    const time_point< _Clock, _Dur > & __t )
```

time\_point\_cast

Definition at line 674 of file chrono.

## 4.17 std::decimal Namespace Reference

## Classes

- class [decimal128](#)
- class [decimal32](#)
- class [decimal64](#)

## Functions

- double **decimal128\_to\_double** ([decimal128](#) \_\_d)
- float **decimal128\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal128\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal128\_to\_long\_long** ([decimal128](#) \_\_d)
- double **decimal32\_to\_double** ([decimal32](#) \_\_d)
- float **decimal32\_to\_float** ([decimal32](#) \_\_d)
- long double **decimal32\_to\_long\_double** ([decimal32](#) \_\_d)
- long long **decimal32\_to\_long\_long** ([decimal32](#) \_\_d)
- double **decimal64\_to\_double** ([decimal64](#) \_\_d)
- float **decimal64\_to\_float** ([decimal64](#) \_\_d)
- long double **decimal64\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal64\_to\_long\_long** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal32](#) \_\_d)
- double **decimal\_to\_double** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal128](#) \_\_d)
- float **decimal\_to\_float** ([decimal32](#) \_\_d)
- float **decimal\_to\_float** ([decimal64](#) \_\_d)
- float **decimal\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)

- static [decimal64](#) **make\_decimal64** (long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32](#) **operator\*** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32](#) **operator\*** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32](#) **operator\*** ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32](#) **operator\*** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)

- `decimal32 operator*` (`decimal32 __lhs`, `long __rhs`)
- `decimal32 operator*` (`decimal32 __lhs`, `long long __rhs`)
- `decimal32 operator*` (`decimal32 __lhs`, `unsigned long long __rhs`)
- `decimal32 operator*` (`int __lhs`, `decimal32 __rhs`)
- `decimal32 operator*` (`unsigned int __lhs`, `decimal32 __rhs`)
- `decimal32 operator*` (`long __lhs`, `decimal32 __rhs`)
- `decimal32 operator*` (`unsigned long __lhs`, `decimal32 __rhs`)
- `decimal32 operator*` (`long long __lhs`, `decimal32 __rhs`)
- `decimal32 operator*` (`unsigned long long __lhs`, `decimal32 __rhs`)
- `decimal64 operator*` (`decimal32 __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `decimal32 __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `int __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `unsigned int __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `long __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `unsigned long __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `long long __rhs`)
- `decimal64 operator*` (`decimal64 __lhs`, `unsigned long long __rhs`)
- `decimal64 operator*` (`int __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`unsigned int __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`long __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`unsigned long __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`long long __lhs`, `decimal64 __rhs`)
- `decimal64 operator*` (`unsigned long long __lhs`, `decimal64 __rhs`)
- `decimal128 operator*` (`decimal32 __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`decimal64 __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `decimal32 __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `decimal64 __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `int __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `unsigned int __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `long __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `unsigned long __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `long long __rhs`)
- `decimal128 operator*` (`decimal128 __lhs`, `unsigned long long __rhs`)
- `decimal128 operator*` (`int __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`unsigned int __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`long __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`unsigned long __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`long long __lhs`, `decimal128 __rhs`)
- `decimal128 operator*` (`unsigned long long __lhs`, `decimal128 __rhs`)
- `decimal32 operator+` (`decimal32 __rhs`)
- `decimal64 operator+` (`decimal64 __rhs`)
- `decimal128 operator+` (`decimal128 __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `decimal32 __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `int __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `unsigned int __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `long __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `unsigned long __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `long long __rhs`)
- `decimal32 operator+` (`decimal32 __lhs`, `unsigned long long __rhs`)

- **decimal32 operator+** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator+** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator+** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator+** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, int \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, long \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator-** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (long \_\_lhs, decimal32 \_\_rhs)

- [decimal32 operator-](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator-](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator-](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator/](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)



- **decimal64 operator/** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator/** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, int \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, long \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- **bool operator<** (int \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)

- bool **operator**< (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal64 \_\_rhs)

- bool **operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator>** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, decimal128 \_\_rhs)

- bool **operator**> (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long long \_\_rhs)

- bool **operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal128 \_\_rhs)

#### 4.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

#### 4.17.2 Function Documentation

##### 4.17.2.1 decimal32\_to\_long\_long()

```
long long std::decimal::decimal32_to_long_long (
    decimal32 __d )
```

Non-conforming extension: Conversion to integral type.

## 4.18 std::placeholders Namespace Reference

### Variables

- const [\\_Placeholder](#)< 1 > [\\_1](#)
- const [\\_Placeholder](#)< 10 > [\\_10](#)
- const [\\_Placeholder](#)< 11 > [\\_11](#)
- const [\\_Placeholder](#)< 12 > [\\_12](#)
- const [\\_Placeholder](#)< 13 > [\\_13](#)
- const [\\_Placeholder](#)< 14 > [\\_14](#)
- const [\\_Placeholder](#)< 15 > [\\_15](#)
- const [\\_Placeholder](#)< 16 > [\\_16](#)
- const [\\_Placeholder](#)< 17 > [\\_17](#)
- const [\\_Placeholder](#)< 18 > [\\_18](#)
- const [\\_Placeholder](#)< 19 > [\\_19](#)
- const [\\_Placeholder](#)< 2 > [\\_2](#)
- const [\\_Placeholder](#)< 20 > [\\_20](#)
- const [\\_Placeholder](#)< 21 > [\\_21](#)
- const [\\_Placeholder](#)< 22 > [\\_22](#)
- const [\\_Placeholder](#)< 23 > [\\_23](#)
- const [\\_Placeholder](#)< 24 > [\\_24](#)
- const [\\_Placeholder](#)< 25 > [\\_25](#)
- const [\\_Placeholder](#)< 26 > [\\_26](#)
- const [\\_Placeholder](#)< 27 > [\\_27](#)
- const [\\_Placeholder](#)< 28 > [\\_28](#)
- const [\\_Placeholder](#)< 29 > [\\_29](#)
- const [\\_Placeholder](#)< 3 > [\\_3](#)
- const [\\_Placeholder](#)< 4 > [\\_4](#)
- const [\\_Placeholder](#)< 5 > [\\_5](#)
- const [\\_Placeholder](#)< 6 > [\\_6](#)
- const [\\_Placeholder](#)< 7 > [\\_7](#)
- const [\\_Placeholder](#)< 8 > [\\_8](#)
- const [\\_Placeholder](#)< 9 > [\\_9](#)

## 4.18.1 Detailed Description

ISO C++11 entities sub-namespace for functional.

## 4.19 std::regex\_constants Namespace Reference

## 5.1 Regular Expression Syntax Options

- enum `__syntax_option` {  
`_S_icase`, `_S_nosubs`, `_S_optimize`, `_S_collate`,  
`_S_ECMAScript`, `_S_basic`, `_S_extended`, `_S_awk`,  
`_S_grep`, `_S_egrep`, `_S_polynomial`, `_S_syntax_last` }
- enum `syntax_option_type` : unsigned int
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type icase`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type nosubs`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type optimize`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type collate`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type ECMAScript`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type basic`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type extended`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type awk`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type grep`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type egrep`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type __polynomial`
- constexpr `syntax_option_type operator&` (`syntax_option_type __a`, `syntax_option_type __b`)
- constexpr `syntax_option_type operator|` (`syntax_option_type __a`, `syntax_option_type __b`)
- constexpr `syntax_option_type operator^` (`syntax_option_type __a`, `syntax_option_type __b`)
- constexpr `syntax_option_type operator~` (`syntax_option_type __a`)
- `syntax_option_type & operator&=` (`syntax_option_type & __a`, `syntax_option_type __b`)
- `syntax_option_type & operator|=` (`syntax_option_type & __a`, `syntax_option_type __b`)
- `syntax_option_type & operator^=` (`syntax_option_type & __a`, `syntax_option_type __b`)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `__match_flag` {  
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,  
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,  
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- enum `match_flag_type` : unsigned int
- `_GLIBCXX17_INLINE` constexpr `match_flag_type match_default`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type match_not_bol`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type match_not_eol`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type match_not_bow`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type match_not_eow`

- `_GLIBCXX17_INLINE constexpr match_flag_type match_any`
- `_GLIBCXX17_INLINE constexpr match_flag_type match_not_null`
- `_GLIBCXX17_INLINE constexpr match_flag_type match_continuous`
- `_GLIBCXX17_INLINE constexpr match_flag_type match_prev_avail`
- `_GLIBCXX17_INLINE constexpr match_flag_type format_default`
- `_GLIBCXX17_INLINE constexpr match_flag_type format_sed`
- `_GLIBCXX17_INLINE constexpr match_flag_type format_no_copy`
- `_GLIBCXX17_INLINE constexpr match_flag_type format_first_only`
- `constexpr match_flag_type operator& (match_flag_type __a, match_flag_type __b)`
- `constexpr match_flag_type operator| (match_flag_type __a, match_flag_type __b)`
- `constexpr match_flag_type operator^ (match_flag_type __a, match_flag_type __b)`
- `constexpr match_flag_type operator~ (match_flag_type __a)`
- `match_flag_type & operator&= (match_flag_type &__a, match_flag_type __b)`
- `match_flag_type & operator|= (match_flag_type &__a, match_flag_type __b)`
- `match_flag_type & operator^= (match_flag_type &__a, match_flag_type __b)`

### 5.3 Error Types

- `enum error_type {`  
`_S_error_collate, _S_error_ctype, _S_error_escape, _S_error_backref,`  
`_S_error_brack, _S_error_paren, _S_error_brace, _S_error_badbrace,`  
`_S_error_range, _S_error_space, _S_error_badrepeat, _S_error_complexity,`  
`_S_error_stack }`
- `constexpr error_type error_collate (_S_error_collate)`
- `constexpr error_type error_ctype (_S_error_ctype)`
- `constexpr error_type error_escape (_S_error_escape)`
- `constexpr error_type error_backref (_S_error_backref)`
- `constexpr error_type error_brack (_S_error_brack)`
- `constexpr error_type error_paren (_S_error_paren)`
- `constexpr error_type error_brace (_S_error_brace)`
- `constexpr error_type error_badbrace (_S_error_badbrace)`
- `constexpr error_type error_range (_S_error_range)`
- `constexpr error_type error_space (_S_error_space)`
- `constexpr error_type error_badrepeat (_S_error_badrepeat)`
- `constexpr error_type error_complexity (_S_error_complexity)`
- `constexpr error_type error_stack (_S_error_stack)`

#### 4.19.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

#### 4.19.2 Enumeration Type Documentation

#### 4.19.2.1 \_\_match\_flag

```
enum std::regex_constants::__match_flag
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 232 of file `regex_constants.h`.

#### 4.19.2.2 \_\_syntax\_option

```
enum std::regex_constants::__syntax_option
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 54 of file `regex_constants.h`.

#### 4.19.2.3 error\_type

```
enum std::regex_constants::error_type
```

The expression contained an invalid collating element name.

Definition at line 49 of file `regex_error.h`.

#### 4.19.2.4 match\_flag\_type

```
enum std::regex_constants::match_flag_type : unsigned int
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 255 of file `regex_constants.h`.



#### 4.19.2.5 syntax\_option\_type

```
enum std::regex_constants::syntax_option_type : unsigned int
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 81 of file `regex_constants.h`.

### 4.19.3 Function Documentation

#### 4.19.3.1 error\_backref()

```
constexpr error_type std::regex_constants::error_backref (
    _S_error_backref )
```

The expression contained an invalid back reference.

#### 4.19.3.2 error\_badbrace()

```
constexpr error_type std::regex_constants::error_badbrace (
    _S_error_badbrace )
```

The expression contained an invalid range in a `{}` expression.

#### 4.19.3.3 error\_badrepeat()

```
constexpr error_type std::regex_constants::error_badrepeat (
    _S_error_badrepeat )
```

One of `*?+{` was not preceded by a valid regular expression.

#### 4.19.3.4 error\_brace()

```
constexpr error_type std::regex_constants::error_brace (
    _S_error_brace )
```

The expression contained mismatched `{` and `}`

#### 4.19.3.5 error\_brack()

```
constexpr error_type std::regex_constants::error_brack (
    _S_error_brack )
```

The expression contained mismatched [ and ].

#### 4.19.3.6 error\_collate()

```
constexpr error_type std::regex_constants::error_collate (
    _S_error_collate )
```

The expression contained an invalid collating element name.

#### 4.19.3.7 error\_complexity()

```
constexpr error_type std::regex_constants::error_complexity (
    _S_error_complexity )
```

The complexity of an attempted match against a regular expression exceeded a pre-set level.

#### 4.19.3.8 error\_ctype()

```
constexpr error_type std::regex_constants::error_ctype (
    _S_error_ctype )
```

The expression contained an invalid character class name.

#### 4.19.3.9 error\_escape()

```
constexpr error_type std::regex_constants::error_escape (
    _S_error_escape )
```

The expression contained an invalid escaped character, or a trailing escape.

#### 4.19.3.10 error\_paren()

```
constexpr error_type std::regex_constants::error_paren (
    _S_error_paren )
```

The expression contained mismatched ( and ).

#### 4.19.3.11 error\_range()

```
constexpr error_type std::regex_constants::error_range (
    _S_error_range )
```

The expression contained an invalid character range, such as [b-a] in most encodings.

#### 4.19.3.12 error\_space()

```
constexpr error_type std::regex_constants::error_space (
    _S_error_space )
```

There was insufficient memory to convert the expression into a finite state machine.

#### 4.19.3.13 error\_stack()

```
constexpr error_type std::regex_constants::error_stack (
    _S_error_stack )
```

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

#### 4.19.3.14 operator&() [1/2]

```
constexpr syntax_option_type std::regex_constants::operator& (
    syntax_option_type __a,
    syntax_option_type __b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 183 of file `regex_constants.h`.

#### 4.19.3.15 operator&() [2/2]

```
constexpr match_flag_type std::regex_constants::operator& (
    match_flag_type __a,
    match_flag_type __b ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 374 of file `regex_constants.h`.

#### 4.19.3.16 operator&=() [1/2]

```
syntax_option_type& std::regex_constants::operator&= (
    syntax_option_type & __a,
    syntax_option_type __b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 208 of file `regex_constants.h`.

#### 4.19.3.17 operator&=() [2/2]

```
match_flag_type& std::regex_constants::operator&= (
    match_flag_type & __a,
    match_flag_type __b ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 399 of file `regex_constants.h`.

#### 4.19.3.18 operator^() [1/2]

```
constexpr syntax_option_type std::regex_constants::operator^ (
    syntax_option_type __a,
    syntax_option_type __b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 197 of file `regex_constants.h`.

**4.19.3.19 operator^()** [2/2]

```
constexpr match_flag_type std::regex_constants::operator^ (
    match_flag_type __a,
    match_flag_type __b ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 388 of file `regex_constants.h`.

**4.19.3.20 operator^=()** [1/2]

```
syntax_option_type& std::regex_constants::operator^= (
    syntax_option_type & __a,
    syntax_option_type __b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 216 of file `regex_constants.h`.

**4.19.3.21 operator^=()** [2/2]

```
match_flag_type& std::regex_constants::operator^= (
    match_flag_type & __a,
    match_flag_type __b ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 407 of file `regex_constants.h`.

#### 4.19.3.22 operator" | () [1/2]

```
constexpr syntax_option_type std::regex_constants::operator| (
    syntax_option_type __a,
    syntax_option_type __b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 190 of file `regex_constants.h`.

#### 4.19.3.23 operator" | () [2/2]

```
constexpr match_flag_type std::regex_constants::operator| (
    match_flag_type __a,
    match_flag_type __b ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 381 of file `regex_constants.h`.

#### 4.19.3.24 operator" |=() [1/2]

```
syntax_option_type& std::regex_constants::operator|= (
    syntax_option_type & __a,
    syntax_option_type __b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 212 of file `regex_constants.h`.

#### 4.19.3.25 operator" | =( ) [2/2]

```
match_flag_type& std::regex_constants::operator|= (
    match_flag_type & __a,
    match_flag_type __b ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 403 of file `regex_constants.h`.

#### 4.19.3.26 operator~() [1/2]

```
constexpr syntax_option_type std::regex_constants::operator~ (
    syntax_option_type __a ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 204 of file `regex_constants.h`.

#### 4.19.3.27 operator~() [2/2]

```
constexpr match_flag_type std::regex_constants::operator~ (
    match_flag_type __a ) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 395 of file `regex_constants.h`.

### 4.19.4 Variable Documentation

#### 4.19.4.1 \_\_polynomial

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::__polynomial
```

Extension: Ensure both space complexity of compiled regex and time complexity execution are not exponential. If specified in a regex with back-references, the exception `regex_constants::error_complexity` will be thrown.

Definition at line 179 of file `regex_constants.h`.

#### 4.19.4.2 awk

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::awk
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\&apos;`, `\&apos;`, and `\ddd` (where `ddd` is one, two, or three octal digits).

Definition at line 152 of file `regex_constants.h`.

#### 4.19.4.3 basic

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::basic
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 132 of file `regex_constants.h`.

#### 4.19.4.4 collate

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::collate
```

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

Definition at line 111 of file `regex_constants.h`.



#### 4.19.4.5 ECMAScript

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::ECMAScript
```

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 122 of file `regex_constants.h`.

#### 4.19.4.6 egrep

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::egrep
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that newlines are treated as whitespace.

Definition at line 170 of file `regex_constants.h`.

#### 4.19.4.7 extended

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::extended
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 141 of file `regex_constants.h`.

#### 4.19.4.8 format\_default

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::format_default
```

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.

- `$`` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in `[1,9]` and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on `[01, 99]`. If `nn <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Definition at line 346 of file `regex_constants.h`.

#### 4.19.4.9 `format_first_only`

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::format_first_only
```

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 370 of file `regex_constants.h`.

Referenced by `std::regex_replace()`.

#### 4.19.4.10 `format_no_copy`

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::format_no_copy
```

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 363 of file `regex_constants.h`.

Referenced by `std::regex_replace()`.

#### 4.19.4.11 `format_sed`

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::format_sed
```

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1-2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 355 of file `regex_constants.h`.

#### 4.19.4.12 `grep`

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::grep
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

Definition at line 161 of file `regex_constants.h`.

#### 4.19.4.13 `icase`

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::icase
```

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 87 of file `regex_constants.h`.

#### 4.19.4.14 `match_any`

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_any
```

If more than one match is possible then any match is an acceptable result.

Definition at line 297 of file `regex_constants.h`.

#### 4.19.4.15 `match_continuous`

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_continuous
```

The expression only matches a sub-sequence that begins at first .

Definition at line 309 of file `regex_constants.h`.

Referenced by `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

#### 4.19.4.16 `match_default`

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_default
```

The default matching rules.

Definition at line 260 of file `regex_constants.h`.

#### 4.19.4.17 match\_not\_bol

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_not_bol
```

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 268 of file regex\_constants.h.

#### 4.19.4.18 match\_not\_bow

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_not_bow
```

The expression \b is not matched against the sub-sequence [first,first).

Definition at line 283 of file regex\_constants.h.

#### 4.19.4.19 match\_not\_eol

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_not_eol
```

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

Definition at line 276 of file regex\_constants.h.

#### 4.19.4.20 match\_not\_eow

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_not_eow
```

The expression \b should not be matched against the sub-sequence [last,last).

Definition at line 290 of file regex\_constants.h.

#### 4.19.4.21 match\_not\_null

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_not_null
```

The expression does not match an empty sequence.

Definition at line 303 of file regex\_constants.h.

Referenced by std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >::operator++().

#### 4.19.4.22 match\_prev\_avail

```
_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::match_prev_avail
```

—first is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the regular expression algorithms 28.11 and iterators 28.12.

Definition at line 317 of file `regex_constants.h`.

#### 4.19.4.23 nosubs

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::nosubs
```

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied `match_results` structure.

Definition at line 95 of file `regex_constants.h`.

#### 4.19.4.24 optimize

```
_GLIBCXX17_INLINE constexpr syntax_option_type std::regex_constants::optimize
```

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 104 of file `regex_constants.h`.

## 4.20 std::rel\_ops Namespace Reference

### Functions

- `template<class _Tp >`  
`bool operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator>= (const _Tp &__x, const _Tp &__y)`

#### 4.20.1 Detailed Description

The generated relational operators are sequestered here.

## 4.20.2 Function Documentation

### 4.20.2.1 operator!=(())

```
template<class _Tp >
bool std::rel_ops::operator!= (
    const _Tp & __x,
    const _Tp & __y ) [inline]
```

Defines != for arbitrary types, in terms of ==.

**Parameters**

$\_x$	A thing.
$\_y$	Another thing.

**Returns**

$\_x \neq \_y$

This function uses `==` to determine its result.

Definition at line 87 of file `stl_relops.h`.

**4.20.2.2 operator<=()**

```
template<class _Tp >
bool std::rel_ops::operator<= (
    const _Tp & __x,
    const _Tp & __y ) [inline]
```

Defines `<=` for arbitrary types, in terms of `<`.

**Parameters**

$\_x$	A thing.
$\_y$	Another thing.

**Returns**

$\_x \leq \_y$

This function uses `<` to determine its result.

Definition at line 113 of file `stl_relops.h`.

**4.20.2.3 operator>()**

```
template<class _Tp >
bool std::rel_ops::operator> (
    const _Tp & __x,
    const _Tp & __y ) [inline]
```

Defines `>` for arbitrary types, in terms of `<`.

## Parameters

$\_x$	A thing.
$\_y$	Another thing.

## Returns

$\_x > \_y$

This function uses `<` to determine its result.

Definition at line 100 of file `stl_relops.h`.

4.20.2.4 `operator>=()`

```
template<class _Tp >
bool std::rel_ops::operator>= (
    const _Tp & __x,
    const _Tp & __y ) [inline]
```

Defines `>=` for arbitrary types, in terms of `<`.

## Parameters

$\_x$	A thing.
$\_y$	Another thing.

## Returns

$\_x >= \_y$

This function uses `<` to determine its result.

Definition at line 126 of file `stl_relops.h`.

4.21 `std::this_thread` Namespace Reference

## Functions

- `void __sleep_for (chrono::seconds, chrono::nanoseconds)`
- `thread::id get_id () noexcept`



- `template<typename _Rep , typename _Period >`  
`void sleep_for (const chrono::duration< _Rep, _Period > &__rtime)`
- `template<typename _Clock , typename _Duration >`  
`void sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)`
- `void yield () noexcept`

#### 4.21.1 Detailed Description

ISO C++ 2011 entities sub-namespace for thread. 30.3.2 Namespace `this_thread`.

#### 4.21.2 Function Documentation

##### 4.21.2.1 `get_id()`

```
thread::id std::this_thread::get_id ( ) [inline], [noexcept]
```

`get_id`

Definition at line 339 of file `thread`.

##### 4.21.2.2 `sleep_for()`

```
template<typename _Rep , typename _Period >  
void std::this_thread::sleep_for (  
    const chrono::duration< _Rep, _Period > & __rtime ) [inline]
```

`sleep_for`

Definition at line 367 of file `thread`.

##### 4.21.2.3 `sleep_until()`

```
template<typename _Clock , typename _Duration >  
void std::this_thread::sleep_until (  
    const chrono::time_point< _Clock, _Duration > & __atime ) [inline]
```

`sleep_until`

Definition at line 389 of file `thread`.

## 4.21.2.4 yield()

```
void std::this_thread::yield ( ) [inline], [noexcept]
```

yield

Definition at line 354 of file thread.

## 4.22 std::tr1 Namespace Reference

## Namespaces

- [\\_\\_detail](#)

## Functions

- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [assoc\\_laguerre](#) (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- float [assoc\\_laguerref](#) (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- long double [assoc\\_laguerrel](#) (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [assoc\\_legendre](#) (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- float [assoc\\_legendref](#) (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- long double [assoc\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tpx , typename \_Tpy >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpx, \_Tpy >::\_\_type [beta](#) (\_Tpx \_\_x, \_Tpy \_\_y)
- float [betaf](#) (float \_\_x, float \_\_y)
- long double [betal](#) (long double \_\_x, long double \_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [comp\\_ellint\\_1](#) (\_Tp \_\_k)
- float [comp\\_ellint\\_1f](#) (float \_\_k)
- long double [comp\\_ellint\\_1l](#) (long double \_\_k)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [comp\\_ellint\\_2](#) (\_Tp \_\_k)
- float [comp\\_ellint\\_2f](#) (float \_\_k)
- long double [comp\\_ellint\\_2l](#) (long double \_\_k)
- template<typename \_Tp , typename \_Tpn >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type [comp\\_ellint\\_3](#) (\_Tp \_\_k, \_Tpn \_\_nu)
- float [comp\\_ellint\\_3f](#) (float \_\_k, float \_\_nu)
- long double [comp\\_ellint\\_3l](#) (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa , typename \_Tpc , typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type [conf\\_hyperg](#) (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float [conf\\_hypergf](#) (float \_\_a, float \_\_c, float \_\_x)
- long double [conf\\_hypergl](#) (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
[std::complex](#)< \_Tp > [conj](#) (const [std::complex](#)< \_Tp > &\_\_z)
- template<typename \_Tp >  
[std::complex](#)< typename \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type > [conj](#) (\_Tp \_\_x)

- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_j (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_k (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_neumann (_Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint\_1 (_Tp __k, _Tpp __phi)`
- `float ellint_1f (float __k, float __phi)`
- `long double ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint\_2 (_Tp __k, _Tpp __phi)`
- `float ellint_2f (float __k, float __phi)`
- `long double ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint\_3 (_Tp __k, _Tpn __nu, _Tpp __phi)`
- `float ellint_3f (float __k, float __nu, float __phi)`
- `long double ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)`
- `float expintf (float __x)`
- `long double expintl (long double __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > fabs (const std::complex< _Tp > &__z)`
- `float fabs (float __x)`
- `long double fabs (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type fabs (_Tp __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)`
- `float laguerref (unsigned int __n, float __x)`
- `long double laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __n, _Tp __x)`

- float **legendref** (unsigned int \_\_n, float \_\_x)
- long double **legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp, typename \_Up >  
std::complex< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **polar** (const \_Tp &\_\_rho, const \_Up &\_\_theta)
- template<typename \_Tp, typename \_Up >  
std::complex< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const std::complex< \_Tp > &\_\_x, const \_Up &\_\_y)
- template<typename \_Tp, typename \_Up >  
std::complex< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const \_Tp &\_\_x, const std::complex< \_Up > &\_\_y)
- template<typename \_Tp, typename \_Up >  
std::complex< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const std::complex< \_Tp > &\_\_x, const std::complex< \_Up > &\_\_y)
- template<typename \_Tp >  
std::complex< \_Tp > **pow** (const std::complex< \_Tp > &\_\_x, const \_Tp &\_\_y)
- template<typename \_Tp >  
std::complex< \_Tp > **pow** (const \_Tp &\_\_x, const std::complex< \_Tp > &\_\_y)
- template<typename \_Tp >  
std::complex< \_Tp > **pow** (const std::complex< \_Tp > &\_\_x, const std::complex< \_Tp > &\_\_y)
- float **pow** (float \_\_x, float \_\_y)
- long double **pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **pow** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **riemann\_zeta** (\_Tp \_\_x)
- float **riemann\_zetaf** (float \_\_x)
- long double **riemann\_zetal** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

#### 4.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

#### 4.22.2 Function Documentation

#### 4.22.2.1 `conf_hyperg()`

```
template<typename _Tpa , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type std::tr1::conf_hyperg (
    _Tpa __a,
    _Tpc __c,
    _Tp __x ) [inline]
```

5.2.1.7 Confluent hypergeometric functions.

Definition at line 1672 of file tr1/cmath.

#### 4.22.2.2 `hyperg()`

```
template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type std::tr1::hyperg (
    _Tpa __a,
    _Tpb __b,
    _Tpc __c,
    _Tp __x ) [inline]
```

5.2.1.17 Hypergeometric functions.

Definition at line 1689 of file tr1/cmath.

### 4.23 `std::tr1::__detail` Namespace Reference

#### 4.23.1 Detailed Description

Implementation details not part of the namespace `std::tr1` interface.

### 4.24 `std::tr2` Namespace Reference

#### Namespaces

- [\\_\\_detail](#)

#### Classes

- struct [\\_\\_dynamic\\_bitset\\_base](#)
- struct [\\_\\_reflection\\_typelist](#)
- struct [\\_\\_reflection\\_typelist<\\_First, \\_Rest... >](#)
- struct [\\_\\_reflection\\_typelist<>](#)
- struct [bases](#)
- class [bool\\_set](#)
- struct [direct\\_bases](#)
- class [dynamic\\_bitset](#)

## Functions

- bool **certainly** (bool\_set \_\_b)
- bool **contains** (bool\_set \_\_s, bool\_set \_\_t)
- bool **equals** (bool\_set \_\_s, bool\_set \_\_t)
- bool **is\_emptyset** (bool\_set \_\_b)
- bool **is\_indeterminate** (bool\_set \_\_b)
- bool **is\_singleton** (bool\_set \_\_b)
- bool\_set **operator!=** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator!=** (bool\_set \_\_s, bool \_\_t)
- bool\_set **operator!=** (bool\_set \_\_s, bool\_set \_\_t)
- bool\_set **operator&** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator&** (bool\_set \_\_s, bool \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc >  
std::basic\_ostream< \_CharT, \_Traits > & **operator<<** (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const  
dynamic\_bitset< \_WordT, \_Alloc > &\_\_x)
- bool\_set **operator==** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator==** (bool\_set \_\_s, bool \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc >  
std::basic\_istream< \_CharT, \_Traits > & **operator>>** (std::basic\_istream< \_CharT, \_Traits > &\_\_is,  
dynamic\_bitset< \_WordT, \_Alloc > &\_\_x)
- bool\_set **operator^** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator^** (bool\_set \_\_s, bool \_\_t)
- bool\_set **operator|** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator|** (bool\_set \_\_s, bool \_\_t)
- bool **possibly** (bool\_set \_\_b)
- bool\_set **set\_complement** (bool\_set \_\_b)
- bool\_set **set\_intersection** (bool \_\_s, bool\_set \_\_t)
- bool\_set **set\_intersection** (bool\_set \_\_s, bool \_\_t)
- bool\_set **set\_intersection** (bool\_set \_\_s, bool\_set \_\_t)
- bool\_set **set\_union** (bool \_\_s, bool\_set \_\_t)
- bool\_set **set\_union** (bool\_set \_\_s, bool \_\_t)
- bool\_set **set\_union** (bool\_set \_\_s, bool\_set \_\_t)
- template<typename \_WordT, typename \_Alloc >  
bool **operator!=** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)
- template<typename \_WordT, typename \_Alloc >  
bool **operator<=** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)
- template<typename \_WordT, typename \_Alloc >  
bool **operator>** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)
- template<typename \_WordT, typename \_Alloc >  
bool **operator>=** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)

- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`

#### 4.24.1 Detailed Description

ISO C++ TR2 entities toplevel namespace is `std::tr2`.

### 4.25 `std::tr2::__detail` Namespace Reference

#### 4.25.1 Detailed Description

Implementation details not part of the namespace `std::tr2` interface.

## 5 Class Documentation

### 5.1 `__cxxabiv1::__forced_unwind` Class Reference

#### 5.1.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

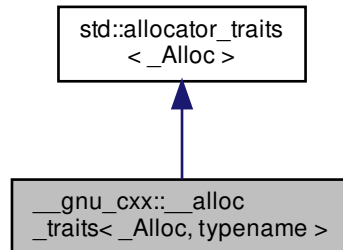
Definition at line 48 of file `cxxabi_forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi\\_forced.h](#)

5.2 `__gnu_cxx::__alloc_traits<_Alloc, typename >` Struct Template Reference

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc, typename >`:



## Public Types

- typedef `std::allocator_traits<_Alloc>_Base_type`
- typedef `_Alloc allocator_type`
- typedef `_Base_type::const_pointer const_pointer`
- typedef `const value_type & const_reference`
- using `const_void_pointer = typename _Ptr<__cv_pointer, const void>::type`
- typedef `_Base_type::difference_type difference_type`
- using `is_always_equal = __detected_or_t< typename is_empty<_Alloc>::type, __equal, _Alloc >`
- typedef `_Base_type::pointer pointer`
- using `propagate_on_container_copy_assignment = __detected_or_t< false_type, __pocca, _Alloc >`
- using `propagate_on_container_move_assignment = __detected_or_t< false_type, __pocma, _Alloc >`
- using `propagate_on_container_swap = __detected_or_t< false_type, __pocs, _Alloc >`
- template<typename \_Tp>  
using `rebind_alloc = __alloc_rebind<_Alloc, _Tp>`
- template<typename \_Tp>  
using `rebind_traits = allocator_traits< rebind_alloc<_Tp> >`
- typedef `value_type & reference`
- typedef `_Base_type::size_type size_type`
- typedef `_Base_type::value_type value_type`
- using `void_pointer = typename _Ptr<__v_pointer, void>::type`

## Static Public Member Functions

- static constexpr bool `_S_always_equal()`
- static constexpr bool `_S_nothrow_move()`
- static void `_S_on_swap(_Alloc &_a, _Alloc &_b)`
- static constexpr bool `_S_propagate_on_copy_assign()`
- static constexpr bool `_S_propagate_on_move_assign()`



- static constexpr bool **\_S\_propagate\_on\_swap** ()
- static \_Alloc **\_S\_select\_on\_copy** (const \_Alloc &\_\_a)
- static pointer allocate (\_Alloc &\_\_a, size\_type \_\_n)
- static pointer allocate (\_Alloc &\_\_a, size\_type \_\_n, const\_void\_pointer \_\_hint)
- static pointer allocate (\_Alloc &\_\_a, size\_type \_\_n)
- static pointer allocate (\_Alloc &\_\_a, size\_type \_\_n, const\_void\_pointer \_\_hint)
- template<typename \_Tp, typename... \_Args>  
static auto **construct** (\_Alloc &\_\_a, \_Tp \*\_\_p, \_Args &&... \_\_args) -> decltype(\_S\_construct(\_\_a, \_\_p, **std::forward**< \_Args >(\_\_args)...))
- template<typename \_Ptr, typename... \_Args>  
static **std::enable\_if**< \_\_is\_custom\_pointer< \_Ptr >::value >::type **construct** (\_Alloc &\_\_a, \_Ptr \_\_p, \_Args &&... \_\_args)
- template<typename \_Tp, typename... \_Args>  
static auto **construct** (\_Alloc &\_\_a, \_Tp \*\_\_p, \_Args &&... \_\_args) -> decltype(\_S\_construct(\_\_a, \_\_p, **std::forward**< \_Args >(\_\_args)...))
- static void **deallocate** (\_Alloc &\_\_a, pointer \_\_p, size\_type \_\_n)
- static void **deallocate** (\_Alloc &\_\_a, pointer \_\_p, size\_type \_\_n)
- template<typename \_Tp >  
static void **destroy** (\_Alloc &\_\_a, \_Tp \*\_\_p)
- template<typename \_Ptr >  
static **std::enable\_if**< \_\_is\_custom\_pointer< \_Ptr >::value >::type **destroy** (\_Alloc &\_\_a, \_Ptr \_\_p)
- template<typename \_Tp >  
static void **destroy** (\_Alloc &\_\_a, \_Tp \*\_\_p)
- static size\_type **max\_size** (const \_Alloc &\_\_a) noexcept
- static size\_type **max\_size** (const \_Alloc &\_\_a) noexcept
- static \_Alloc **select\_on\_container\_copy\_construction** (const \_Alloc &\_\_rhs)

## Protected Types

- template<typename \_Tp >  
using **\_\_c\_pointer** = typename \_Tp::const\_pointer
- template<typename \_Tp >  
using **\_\_cv\_pointer** = typename \_Tp::const\_void\_pointer
- template<typename \_Tp >  
using **\_\_equal** = typename \_Tp::is\_always\_equal
- template<typename \_Tp >  
using **\_\_pocca** = typename \_Tp::propagate\_on\_container\_copy\_assignment
- template<typename \_Tp >  
using **\_\_pocma** = typename \_Tp::propagate\_on\_container\_move\_assignment
- template<typename \_Tp >  
using **\_\_pocs** = typename \_Tp::propagate\_on\_container\_swap
- template<typename \_Tp >  
using **\_\_pointer** = typename \_Tp::pointer
- template<typename \_Tp >  
using **\_\_v\_pointer** = typename \_Tp::void\_pointer

### 5.2.1 Detailed Description

```
template<typename _Alloc, typename = typename _Alloc::value_type>
struct __gnu_cxx::__alloc_traits< _Alloc, typename >
```

Uniform interface to C++98 and C++11 allocators.

Definition at line 50 of file ext/alloc\_traits.h.

### 5.2.2 Member Typedef Documentation

#### 5.2.2.1 `const_void_pointer`

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer = typename _Ptr<__cv_pointer, const
void>::type [inherited]
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 151 of file `bits/alloc_traits.h`.

#### 5.2.2.2 `is_always_equal`

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal = __detected_or_t<typename is_empty<_↵
Alloc>::type, __equal, _Alloc> [inherited]
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

Definition at line 203 of file `bits/alloc_traits.h`.

#### 5.2.2.3 `propagate_on_container_copy_assignment`

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment = __detected_or_↵
t<false_type, __pocca, _Alloc> [inherited]
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 176 of file `bits/alloc_traits.h`.

#### 5.2.2.4 propagate\_on\_container\_move\_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment = __detected_or_t<false_type, __pocma, _Alloc> [inherited]
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 185 of file `bits/alloc_traits.h`.

#### 5.2.2.5 propagate\_on\_container\_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap = __detected_or_t<false_type, __pocs, _Alloc> [inherited]
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 194 of file `bits/alloc_traits.h`.

#### 5.2.2.6 void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer = typename _Ptr<__v_pointer, void>::type
[inherited]
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 143 of file `bits/alloc_traits.h`.

### 5.2.3 Member Function Documentation

#### 5.2.3.1 allocate() [1/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static pointer std::allocator_traits< _Alloc >::allocate [inline], [static]
```

Allocate memory.

## Parameters

<code>↔ _a</code>	An allocator.
<code>↔ _n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 300 of file `bits/alloc_traits.h`.

5.2.3.2 `allocate()` [2/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static pointer std::allocator_traits< _Alloc >::allocate [inline], [static]
```

Allocate memory.

## Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

## Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 315 of file `bits/alloc_traits.h`.

5.2.3.3 `allocate()` [3/4]

```
template<typename _Alloc>
static pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n ) [inline], [static], [inherited]
```

Allocate memory.

## Parameters

<code>↔ _a</code>	An allocator.
<code>↔ _n</code>	The number of objects to allocate space for.
Generated by Doxygen	

Calls `a.allocate(n)`

Definition at line 300 of file `bits/alloc_traits.h`.

Referenced by `std::__allocate_guarded()`.

#### 5.2.3.4 `allocate()` [4/4]

```
template<typename _Alloc>
static pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n,
    const_void_pointer __hint ) [inline], [static], [inherited]
```

Allocate memory.

##### Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

##### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 315 of file `bits/alloc_traits.h`.

#### 5.2.3.5 `construct()` [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp , typename... _Args>
static auto std::allocator_traits< _Alloc >::construct (
    typename _Tp ,
    typename... _Args ) [inline], [static]
```

Construct an object of type `_Tp`.

##### Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args) ...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 342 of file `bits/alloc_traits.h`.

#### 5.2.3.6 `construct()` [2/2]

```
template<typename _Alloc>
template<typename _Tp, typename... _Args>
static auto std::allocator_traits<_Alloc>::construct (
    _Alloc & __a,
    _Tp * __p,
    _Args &&... __args ) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__↵
args)...))    [inline], [static], [inherited]
```

Construct an object of type `_Tp`.

##### Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args) ...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 342 of file `bits/alloc_traits.h`.

#### 5.2.3.7 `deallocate()` [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static void std::allocator_traits<_Alloc>::deallocate [inline], [static]
```

Deallocate memory.

##### Parameters

<code>↵ __a</code>	An allocator.
<code>↵ __p</code>	Pointer to the memory to deallocate.
<code>↵ __n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 327 of file bits/alloc\_traits.h.

### 5.2.3.8 deallocate() [2/2]

```
template<typename _Alloc>
static void std::allocator_traits< _Alloc >::deallocate (
    _Alloc & __a,
    pointer __p,
    size_type __n ) [inline], [static], [inherited]
```

Deallocate memory.

#### Parameters

$\hookleftarrow$ __a	An allocator.
$\hookleftarrow$ __p	Pointer to the memory to deallocate.
$\hookleftarrow$ __n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 327 of file bits/alloc\_traits.h.

Referenced by `std::__allocated_ptr< _Alloc >::~~__allocated_ptr()`.

### 5.2.3.9 destroy() [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp >
static void std::allocator_traits< _Alloc >::destroy (
    typename _Tp ) [inline], [static]
```

Destroy an object of type `_Tp`.

#### Parameters

$\hookleftarrow$ __a	An allocator.
$\hookleftarrow$ __p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 355 of file bits/alloc\_traits.h.

**5.2.3.10 `destroy()`** [2/2]

```
template<typename _Alloc>
template<typename _Tp >
static void std::allocator_traits< _Alloc >::destroy (
    _Alloc & __a,
    _Tp * __p ) [inline], [static], [inherited]
```

Destroy an object of type `_Tp`.

**Parameters**

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 355 of file `bits/alloc_traits.h`.

**5.2.3.11 `max_size()`** [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static size_type std::allocator_traits< _Alloc >::max_size [inline], [static], [noexcept]
```

The maximum supported allocation size.

**Parameters**

<code>__a</code>	An allocator.
------------------	---------------

**Returns**

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 366 of file `bits/alloc_traits.h`.



### 5.2.3.12 `max_size()` [2/2]

```
template<typename _Alloc>
static size_type std::allocator_traits< _Alloc >::max_size (
    const _Alloc & __a ) [inline], [static], [noexcept], [inherited]
```

The maximum supported allocation size.

#### Parameters

<code>__a</code>	An allocator.
------------------	---------------

#### Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 366 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::max_size()`, and `std::list< __inp, __rebind_inp >::max_size()`.

### 5.2.3.13 `select_on_container_copy_construction()`

```
template<typename _Alloc>
static _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
    const _Alloc & __rhs ) [inline], [static], [inherited]
```

Obtain an allocator to use when copying a container.

#### Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

#### Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 378 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [ext/alloc\\_traits.h](#)

### 5.3 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference

Inherits `__gnu_cxx::__common_pool_base<_PoolTp, _Thread >`.

#### 5.3.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy<_PoolTp, _Thread >
```

Policy for shared `__pool` objects.

Definition at line 460 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

### 5.4 `__gnu_cxx::__detail::__mini_vector<_Tp >` Class Template Reference

#### Public Types

- typedef const `_Tp` & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef size\_t **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator `__pos`) throw ()
- void **insert** (iterator `__pos`, const\_reference `__x`)
- reference **operator[]** (const size\_type `__pos`) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference `__x`)
- size\_type **size** () const throw ()

#### 5.4.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::__mini_vector< _Tp >
```

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses operator `new()` to get memory, and operator `delete()` to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 70 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

### 5.5 `__gnu_cxx::__detail::Bitmap_counter< _Tp >` Class Template Reference

#### Public Member Functions

- `Bitmap_counter` ([\\_BPVector](#) &Rvbp, long \_\_index=-1)
- pointer `_M_base` () const throw ()
- bool `_M_finished` () const throw ()
- `size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- void `_M_reset` (long \_\_index=-1) throw ()
- void `_M_set_internal_bitmap` (`size_t * __new_internal_marker`) throw ()
- `_Index_type _M_where` () const throw ()
- [\\_Bitmap\\_counter](#) & `operator++` () throw ()

#### 5.5.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::Bitmap_counter< _Tp >
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

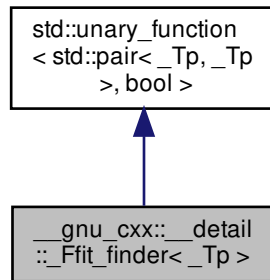
Definition at line 397 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

5.6 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder<_Tp>`:



## Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

## Public Member Functions

- `size_t * _M_get () const throw ()`
- `_Counter_type _M_offset () const throw ()`
- `bool operator() (_Block_pair __bp) throw ()`

## 5.6.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder<_Tp>
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 332 of file `bitmap_allocator.h`.

## 5.6.2 Member Typedef Documentation

### 5.6.2.1 argument\_type

```
typedef std::pair< _Tp, _Tp > std::unary_function< std::pair< _Tp, _Tp > , bool >::argument_type
[inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

### 5.6.2.2 result\_type

```
typedef bool std::unary_function< std::pair< _Tp, _Tp > , bool >::result_type [inherited]
```

result\_type is the return type

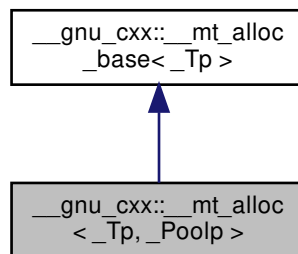
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.7 \_\_gnu\_cxx::\_\_mt\_alloc< \_Tp, \_Poolp > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_mt\_alloc< \_Tp, \_Poolp >:



### Public Types

- typedef \_Poolp **\_\_policy\_type**
- typedef \_Poolp::pool\_type **\_\_pool\_type**
- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- `__mt_alloc` (const `__mt_alloc` &) noexcept
- `template<typename _Tp1, typename _Poolp1 >`  
`__mt_alloc` (const `__mt_alloc<_Tp1, _Poolp1>` &) noexcept
- `const __pool_base::Tune __M_get_options` ()
- `void __M_set_options` (`__pool_base::Tune` \_\_t)
- `pointer address` (reference \_\_x) const noexcept
- `const_pointer address` (const\_reference \_\_x) const noexcept
- `pointer allocate` (size\_type \_\_n, const void \*=0)
- `template<typename _Up, typename... _Args>`  
`void construct` (`_Up *`\_\_p, `_Args` &&... \_\_args)
- `void deallocate` (pointer \_\_p, size\_type \_\_n)
- `template<typename _Up >`  
`void destroy` (`_Up *`\_\_p)
- `size_type max_size` () const noexcept

## 5.7.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc<_Tp, _Poolp>
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt\\_allocator.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt_allocator.html).

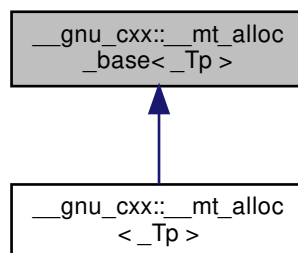
Definition at line 639 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

5.8 `__gnu_cxx::__mt_alloc_base<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc_base<_Tp>`:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

#### 5.8.1 Detailed Description

```
template<typename _Tp>  
class __gnu_cxx::__mt_alloc_base<_Tp >
```

Base class for \_Tp dependent member functions.

Definition at line 570 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.9 \_\_gnu\_cxx::\_\_per\_type\_pool\_policy<\_Tp, \_PoolTp, \_Thread > Struct Template Reference

Inherits [\\_\\_gnu\\_cxx::\\_\\_per\\_type\\_pool\\_base<\\_Tp, \\_PoolTp, \\_Thread >](#).

#### 5.9.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread>  
struct __gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread >
```

Policy for individual \_\_pool objects.

Definition at line 555 of file mt\_allocator.h.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.10 \_\_gnu\_cxx::\_\_pool&lt;\_Thread &gt; Class Template Reference

## 5.10.1 Detailed Description

```
template<bool _Thread>
class __gnu_cxx::__pool<_Thread >
```

Data describing the underlying memory pool, parameterized on threading support.

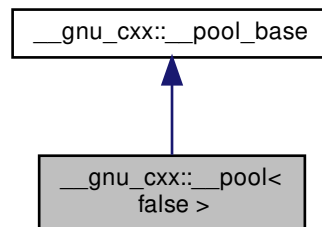
Definition at line 192 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.11 \_\_gnu\_cxx::\_\_pool&lt;false &gt; Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool<false >`:



## Public Types

- typedef unsigned short int **\_Binmap\_type**

## Public Member Functions

- **\_\_pool** (const \_\_pool\_base:: \_Tune & \_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record &, \_Block\_record \*, size\_t)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- void **\_M\_destroy** () throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \* \_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)



### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

#### 5.11.1 Detailed Description

```
template<>
class __gnu_cxx::__pool< false >
```

Specialization for single thread.

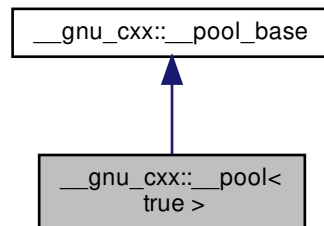
Definition at line 196 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 5.12 `__gnu_cxx::__pool< true >` Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool< true >`:



### Public Types

- `typedef unsigned short int _Binmap_type`

## Public Member Functions

- **\_\_pool** (const \_\_pool\_base::\_Tune &\_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record &\_\_bin, \_Block\_record \*\_\_block, size\_t \_\_thread\_id)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- void **\_M\_destroy** () throw ()
- void **\_M\_destroy\_thread\_key** (void \*) throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize** (\_\_destroy\_handler)
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \*\_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)

## Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

## 5.12.1 Detailed Description

```
template<>
```

```
class __gnu_cxx::__pool< true >
```

Specialization for thread enabled, via gthreads.h.

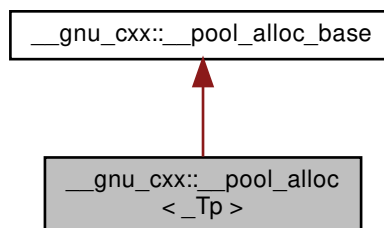
Definition at line 263 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.13 \_\_gnu\_cxx::\_\_pool\_alloc&lt;\_Tp&gt; Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc<\_Tp>:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#) &) noexcept
- template<typename \_Tp1 >  
**\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#)< \_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*==0)
- template<typename \_Up , typename... \_Args>  
void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up >  
void **destroy** (\_Up \* \_\_p)
- size\_type **max\_size** () const noexcept

### Private Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

### Private Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

### Static Private Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

## 5.13.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__pool_alloc< _Tp >
```

Allocator using a memory pool with a single lock.

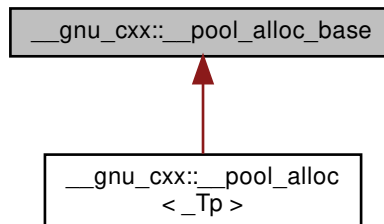
Definition at line 126 of file pool\_allocator.h.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 5.14 \_\_gnu\_cxx::\_\_pool\_alloc\_base Class Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc\_base:



## Protected Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

## Protected Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

## Static Protected Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

### 5.14.1 Detailed Description

Base class for `__pool_alloc`.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size  $> \_S\_max\_bytes$ , the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `\_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

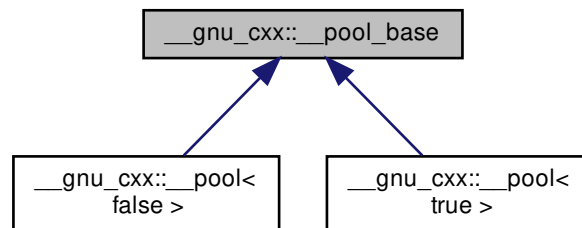
Definition at line 78 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 5.15 `__gnu_cxx::__pool_base` Struct Reference

Inheritance diagram for `__gnu_cxx::__pool_base`:



## Public Types

- typedef unsigned short int **\_Binmap\_type**

### Public Member Functions

- `__pool_base` (const `_Tune` &\_\_options)
- `bool _M_check_threshold` (size\_t \_\_bytes)
- `size_t _M_get_align` ()
- `size_t _M_get_binmap` (size\_t \_\_bytes)
- `const _Tune &_M_get_options` () const
- `void _M_set_options` (\_Tune \_\_t)

### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

#### 5.15.1 Detailed Description

Base class for pool object.

Definition at line 51 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.16 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility<_CharT, _Traits, _Alloc>`.

### Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility<_CharT, _Traits, _Alloc> _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

## Public Member Functions

- `__rc_string_base` (const `_Alloc` & `_a`)
- `__rc_string_base` (const `__rc_string_base` & `__rcs`)
- `__rc_string_base` (`__rc_string_base` && `__rcs`)
- `__rc_string_base` (size\_type `__n`, `_CharT` `__c`, const `_Alloc` & `_a`)
- `template<typename _InputIterator >`  
`__rc_string_base` (`_InputIterator` `__beg`, `_InputIterator` `__end`, const `_Alloc` & `_a`)
- `void` `_M_assign` (const `__rc_string_base` & `__rcs`)
- `size_type` `_M_capacity` () const
- `void` `_M_clear` ()
- `bool` `_M_compare` (const `__rc_string_base` &) const
- `template<>`  
`bool` `_M_compare` (const `__rc_string_base` & `__rcs`) const
- `template<>`  
`bool` `_M_compare` (const `__rc_string_base` & `__rcs`) const
- `_CharT *` `_M_data` () const
- `void` `_M_erase` (size\_type `__pos`, size\_type `__n`)
- `allocator_type` & `_M_get_allocator` ()
- `const allocator_type` & `_M_get_allocator` () const
- `bool` `_M_is_shared` () const
- `void` `_M_leak` ()
- `size_type` `_M_length` () const
- `size_type` `_M_max_size` () const
- `void` `_M_mutate` (size\_type `__pos`, size\_type `__len1`, const `_CharT *` `__s`, size\_type `__len2`)
- `void` `_M_reserve` (size\_type `__res`)
- `void` `_M_set_leaked` ()
- `void` `_M_set_length` (size\_type `__n`)
- `void` `_M_swap` (`__rc_string_base` & `__rcs`)
- `template<typename _InIterator >`  
`_CharT *` `_S_construct` (`_InIterator` `__beg`, `_InIterator` `__end`, const `_Alloc` & `_a`, [std::forward\\_iterator\\_tag](#))

## Protected Types

- `typedef` `__gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > >` `__const_rc_iterator`
- `typedef` `__gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, ↵__sso_string_base > >` `__const_sso_iterator`
- `typedef` `__gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > >` `__rc_iterator`
- `typedef` `__gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, ↵__sso_string_base > >` `__sso_iterator`
- `typedef` `_CharT_alloc_type::const_pointer` `const_pointer`
- `typedef` `_CharT_alloc_type::difference_type` `difference_type`
- `typedef` `_CharT_alloc_type::pointer` `pointer`

## Static Protected Member Functions

- static void `_S_assign` (`_CharT * __d`, `size_type __n`, `_CharT __c`)
- static int `_S_compare` (`size_type __n1`, `size_type __n2`)
- static void `_S_copy` (`_CharT * __d`, `const _CharT * __s`, `size_type __n`)
- template<typename `_Iterator`>  
static void `_S_copy_chars` (`_CharT * __p`, `_Iterator __k1`, `_Iterator __k2`)
- static void `_S_copy_chars` (`_CharT * __p`, `__sso_iterator __k1`, `__sso_iterator __k2`)
- static void `_S_copy_chars` (`_CharT * __p`, `__const_sso_iterator __k1`, `__const_sso_iterator __k2`)
- static void `_S_copy_chars` (`_CharT * __p`, `__rc_iterator __k1`, `__rc_iterator __k2`)
- static void `_S_copy_chars` (`_CharT * __p`, `__const_rc_iterator __k1`, `__const_rc_iterator __k2`)
- static void `_S_copy_chars` (`_CharT * __p`, `_CharT * __k1`, `_CharT * __k2`)
- static void `_S_copy_chars` (`_CharT * __p`, `const _CharT * __k1`, `const _CharT * __k2`)
- static void `_S_move` (`_CharT * __d`, `const _CharT * __s`, `size_type __n`)

## 5.16.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>
```

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[ __rc_string_base<char_type>]      _M_capacity
_M_datapointer                    _M_refcount
_M_p ----->                    unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)



## 5.17 `__gnu_cxx::__scoped_lock` Class Reference

### Public Types

- typedef `__mutex` **`__mutex_type`**

### Public Member Functions

- **`__scoped_lock`** (`__mutex_type` &`__name`)

#### 5.17.1 Detailed Description

Scoped lock idiom.

Definition at line 231 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 5.18 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >` Class Template Reference

Inherits `_Base<_CharT, _Traits, _Alloc >`.

### Public Types

- typedef `_Alloc` **`allocator_type`**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` **`const_iterator`**
- typedef `_CharT_alloc_type::const_pointer` **`const_pointer`**
- typedef `const value_type &` **`const_reference`**
- typedef `std::reverse_iterator< const_iterator >` **`const_reverse_iterator`**
- typedef `_CharT_alloc_type::difference_type` **`difference_type`**
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` **`iterator`**
- typedef `_CharT_alloc_type::pointer` **`pointer`**
- typedef `value_type &` **`reference`**
- typedef `std::reverse_iterator< iterator >` **`reverse_iterator`**
- typedef `_CharT_alloc_type::size_type` **`size_type`**
- typedef `_Traits` **`traits_type`**
- typedef `_Traits::char_type` **`value_type`**

## Public Member Functions

- `__versa_string` (const `_Alloc` & `a`=`_Alloc`()) noexcept
- `__versa_string` (const `__versa_string` & `str`)
- `__versa_string` (`__versa_string` && `str`) noexcept
- `__versa_string` (std::initializer\_list< `_CharT` > `l`, const `_Alloc` & `a`=`_Alloc`())
- `__versa_string` (const `__versa_string` & `str`, size\_type `pos`, size\_type `n`=npos)
- `__versa_string` (const `__versa_string` & `str`, size\_type `pos`, size\_type `n`, const `_Alloc` & `a`)
- `__versa_string` (const `_CharT` \* `s`, size\_type `n`, const `_Alloc` & `a`=`_Alloc`())
- `__versa_string` (const `_CharT` \* `s`, const `_Alloc` & `a`=`_Alloc`())
- `__versa_string` (size\_type `n`, `_CharT` `c`, const `_Alloc` & `a`=`_Alloc`())
- template<class `_InputIterator` , typename = std::RequireInputIter< `_InputIterator` >>  
`__versa_string` (`_InputIterator` `beg`, `_InputIterator` `end`, const `_Alloc` & `a`=`_Alloc`())
- `~__versa_string` () noexcept
- template<typename `_InputIterator` >  
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > & **`M_replace_dispatch`** (const\_iterator `i1`, const\_iterator `i2`, `_InputIterator` `k1`, `_InputIterator` `k2`, std::false\_type)
- `__versa_string` & `append` (const `__versa_string` & `str`)
- `__versa_string` & `append` (const `__versa_string` & `str`, size\_type `pos`, size\_type `n`)
- `__versa_string` & `append` (const `_CharT` \* `s`, size\_type `n`)
- `__versa_string` & `append` (const `_CharT` \* `s`)
- `__versa_string` & `append` (size\_type `n`, `_CharT` `c`)
- `__versa_string` & `append` (std::initializer\_list< `_CharT` > `l`)
- template<class `_InputIterator` , typename = std::RequireInputIter< `_InputIterator` >>  
`__versa_string` & `append` (`_InputIterator` `first`, `_InputIterator` `last`)
- `__versa_string` & `assign` (const `__versa_string` & `str`)
- `__versa_string` & `assign` (`__versa_string` && `str`) noexcept
- `__versa_string` & `assign` (const `__versa_string` & `str`, size\_type `pos`, size\_type `n`)
- `__versa_string` & `assign` (const `_CharT` \* `s`, size\_type `n`)
- `__versa_string` & `assign` (const `_CharT` \* `s`)
- `__versa_string` & `assign` (size\_type `n`, `_CharT` `c`)
- template<class `_InputIterator` , typename = std::RequireInputIter< `_InputIterator` >>  
`__versa_string` & `assign` (`_InputIterator` `first`, `_InputIterator` `last`)
- `__versa_string` & `assign` (std::initializer\_list< `_CharT` > `l`)
- const\_reference `at` (size\_type `n`) const
- reference `at` (size\_type `n`)
- reference `back` () noexcept
- const\_reference `back` () const noexcept
- iterator `begin` () noexcept
- const\_iterator `begin` () const noexcept
- const `_CharT` \* `c_str` () const noexcept
- size\_type `capacity` () const noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- int `compare` (const `__versa_string` & `str`) const
- int `compare` (size\_type `pos`, size\_type `n`, const `__versa_string` & `str`) const
- int `compare` (size\_type `pos1`, size\_type `n1`, const `__versa_string` & `str`, size\_type `pos2`, size\_type `n2`) const
- int `compare` (const `_CharT` \* `s`) const
- int `compare` (size\_type `pos`, size\_type `n1`, const `_CharT` \* `s`) const

- `int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const`
- `size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `const _CharT * data () const noexcept`
- `bool empty () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `__versa_string & erase (size_type __pos=0, size_type __n=npos)`
- `iterator erase (const_iterator __position)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `size_type find (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type find (const __versa_string & __str, size_type __pos=0) const noexcept`
- `size_type find (const _CharT *__s, size_type __pos=0) const`
- `size_type find (_CharT __c, size_type __pos=0) const noexcept`
- `size_type find_first_not_of (const __versa_string & __str, size_type __pos=0) const noexcept`
- `size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const`
- `size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept`
- `size_type find_first_of (const __versa_string & __str, size_type __pos=0) const noexcept`
- `size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type find_first_of (const _CharT *__s, size_type __pos=0) const`
- `size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept`
- `size_type find_last_not_of (const __versa_string & __str, size_type __pos=npos) const noexcept`
- `size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const`
- `size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept`
- `size_type find_last_of (const __versa_string & __str, size_type __pos=npos) const noexcept`
- `size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const`
- `size_type find_last_of (const _CharT *__s, size_type __pos=npos) const`
- `size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept`
- `reference front () noexcept`
- `const_reference front () const noexcept`
- `allocator_type get_allocator () const noexcept`
- `iterator insert (const_iterator __p, size_type __n, _CharT __c)`
- `template<class _InputIterator, typename = std::::RequireInputIter<_InputIterator>>>`  
`iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)`
- `iterator insert (const_iterator __p, std::initializer_list<_CharT> __l)`
- `__versa_string & insert (size_type __pos1, const __versa_string & __str)`
- `__versa_string & insert (size_type __pos1, const __versa_string & __str, size_type __pos2, size_type __n)`
- `__versa_string & insert (size_type __pos, const _CharT *__s, size_type __n)`
- `__versa_string & insert (size_type __pos, const _CharT *__s)`
- `__versa_string & insert (size_type __pos, size_type __n, _CharT __c)`
- `iterator insert (const_iterator __p, _CharT __c)`
- `size_type length () const noexcept`
- `size_type max_size () const noexcept`
- `__versa_string & operator+= (const __versa_string & __str)`
- `__versa_string & operator+= (const _CharT *__s)`
- `__versa_string & operator+= (_CharT __c)`
- `__versa_string & operator+= (std::initializer_list<_CharT> __l)`
- `__versa_string & operator= (const __versa_string & __str)`

- `__versa_string & operator= ( __versa_string &&__str )` noexcept
- `__versa_string & operator= ( std::initializer_list<_CharT> __l )`
- `__versa_string & operator= ( const _CharT *__s )`
- `__versa_string & operator= ( _CharT __c )`
- `const_reference operator[] ( size_type __pos )` const noexcept
- `reference operator[] ( size_type __pos )` noexcept
- `void pop_back ()`
- `void push_back ( _CharT __c )`
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rend ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- `__versa_string & replace ( size_type __pos, size_type __n, const __versa_string &__str )`
- `__versa_string & replace ( size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2 )`
- `__versa_string & replace ( size_type __pos, size_type __n1, const _CharT *__s, size_type __n2 )`
- `__versa_string & replace ( size_type __pos, size_type __n1, const _CharT *__s )`
- `__versa_string & replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, const __versa_string &__str )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, const _CharT *__s, size_type __n )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, const _CharT *__s )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c )`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`__versa_string & replace ( const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2 )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, _CharT *__k1, _CharT *__k2 )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, const _CharT *__k1, const _CharT *__k2 )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, iterator __k1, iterator __k2 )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, const_iterator __k1, const_iterator __k2 )`
- `__versa_string & replace ( const_iterator __i1, const_iterator __i2, std::initializer_list<_CharT> __l )`
- `void reserve ( size_type __res_arg=0 )`
- `void resize ( size_type __n, _CharT __c )`
- `void resize ( size_type __n )`
- `size_type rfind ( const __versa_string &__str, size_type __pos=npow )` const noexcept
- `size_type rfind ( const _CharT *__s, size_type __pos, size_type __n )` const
- `size_type rfind ( const _CharT *__s, size_type __pos=npow )` const
- `size_type rfind ( _CharT __c, size_type __pos=npow )` const noexcept
- `void shrink_to_fit ()` noexcept
- `size_type size ()` const noexcept
- `__versa_string substr ( size_type __pos=0, size_type __n=npow )` const
- `void swap ( __versa_string &__s )` noexcept

#### Static Public Attributes

- static const size\_type `npos`

### 5.18.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >
```

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

Definition at line 56 of file `vstring.h`.

### 5.18.2 Constructor & Destructor Documentation

#### 5.18.2.1 `__versa_string()` [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const _Alloc & __a = _Alloc() ) [inline], [explicit], [noexcept]
```

Construct an empty string using allocator `a`.

Definition at line 137 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr()`.

#### 5.18.2.2 `__versa_string()` [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Construct string with copy of value of `__str`.

#### Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 145 of file `vstring.h`.

5.18.2.3 `__versa_string()` [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    __versa_string< _CharT, _Traits, _Alloc, _Base > && __str ) [inline], [noexcept]
```

String move constructor.

## Parameters

<code>__str</code>	Source string.
--------------------	----------------

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

Definition at line 157 of file `vstring.h`.

5.18.2.4 `__versa_string()` [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    std::initializer_list< _CharT > __l,
    const _Alloc & __a = _Alloc() ) [inline]
```

Construct string from an initializer list.

## Parameters

<code>__l</code>	<code>std::initializer_list</code> of characters.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 165 of file `vstring.h`.

5.18.2.5 `__versa_string()` [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n = npos ) [inline]
```

Construct string as copy of a substring.

## Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy (default remainder).

Definition at line 176 of file `vstring.h`.

5.18.2.6 `__versa_string()` [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n,
    const _Alloc & __a ) [inline]
```

Construct string as copy of a substring.

## Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.

Definition at line 191 of file `vstring.h`.

5.18.2.7 `__versa_string()` [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const _CharT * __s,
    size_type __n,
    const _Alloc & __a = _Alloc() ) [inline]
```

Construct string initialized by a character array.

## Parameters

<code>__s</code>	Source character array.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 208 of file `vstring.h`.

#### 5.18.2.8 `__versa_string()` [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    const _CharT * __s,
    const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as copy of a C string.

##### Parameters

<code>__s</code>	Source C string.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 217 of file `vstring.h`.

#### 5.18.2.9 `__versa_string()` [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    size_type __n,
    _CharT __c,
    const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as multiple characters.

##### Parameters

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 227 of file `vstring.h`.



5.18.2.10 `__versa_string()` [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
    _InputIterator __beg,
    _InputIterator __end,
    const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as copy of a range.

## Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 242 of file `vstring.h`.

5.18.2.11 `~__versa_string()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~__versa_string ( ) [inline], [noexcept]
```

Destroy the string instance.

Definition at line 249 of file `vstring.h`.

## 5.18.3 Member Function Documentation

5.18.3.1 `append()` [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Append a string to this string.

## Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

**Returns**

Reference to this string.

Definition at line 692 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=()`.

**5.18.3.2 `append()`** [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n ) [inline]
```

Append a substring.

**Parameters**

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 709 of file `vstring.h`.

**5.18.3.3 `append()`** [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    const _CharT * __s,
    size_type __n ) [inline]
```

Append a C substring.

#### Parameters

<code>__s</code>	The C string to append.
<code>__n</code>	The number of characters to append.

#### Returns

Reference to this string.

Definition at line 721 of file `vstring.h`.

#### 5.18.3.4 `append()` [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    const _CharT * __s ) [inline]
```

Append a C string.

#### Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

#### Returns

Reference to this string.

Definition at line 734 of file `vstring.h`.

#### 5.18.3.5 `append()` [5/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    size_type __n,
    _CharT __c ) [inline]
```

Append multiple characters.

**Parameters**

<code>__n</code>	The number of characters to append.
<code>__c</code>	The character to use.

**Returns**

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 751 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.18.3.6 `append()`** [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (
    std::initializer_list<_CharT > __l ) [inline]
```

Append an `initializer_list` of characters.

**Parameters**

<code>__l</code>	The <code>initializer_list</code> of characters to append.
------------------	--

**Returns**

Reference to this string.

Definition at line 761 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

### 5.18.3.7 `append()` [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Append a range of characters.

#### Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

#### Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 780 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

### 5.18.3.8 `assign()` [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Set value to contents of another string.

#### Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

#### Returns

Reference to this string.

Definition at line 803 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator=()`.

5.18.3.9 `assign()` [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    __versa_string< _CharT, _Traits, _Alloc, _Base > && __str ) [inline], [noexcept]
```

Set value to contents of another string.

## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

## Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 819 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

5.18.3.10 `assign()` [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos,
    size_type __n ) [inline]
```

Set value to a substring of a string.

## Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 840 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

5.18.3.11 `assign()` [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const _CharT * __s,
    size_type __n ) [inline]
```

Set value to a C substring.

## Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

## Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 857 of file `vstring.h`.

5.18.3.12 `assign()` [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    const _CharT * __s ) [inline]
```

Set value to contents of a C string.

## Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

## Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 873 of file `vstring.h`.

5.18.3.13 `assign()` [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
    size_type __n,
    _CharT __c ) [inline]
```

Set value to multiple characters.

## Parameters

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

## Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 890 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.



5.18.3.14 `assign()` [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Set value to a range of characters.

## Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

## Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 909 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

5.18.3.15 `assign()` [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
    std::initializer_list< _CharT > __l ) [inline]
```

Set value to an `initializer_list` of characters.

## Parameters

<code>↔</code>	The <code>initializer_list</code> of characters to assign.
<code>↔</code>	
<code>↔</code>	
<code>↔</code>	
<code>/</code>	

## Returns

Reference to this string.

Definition at line 919 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

#### 5.18.3.16 `at()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (
    size_type __n ) const [inline]
```

Provides access to the data contained in the string.

##### Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

##### Returns

Read-only (const) reference to the character.

##### Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 577 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

#### 5.18.3.17 `at()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (
    size_type __n ) [inline]
```

Provides access to the data contained in the string.

**Parameters**

<code>_↔</code>	The index of the character to access.
<code>_n</code>	

**Returns**

Read/write reference to the character.

**Exceptions**

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 599 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.18.3.18 back()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the string.

Definition at line 632 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_↔string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.18.3.19 back()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 640 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_↔string<_CharT, _Traits, _Alloc, _Base>::size()`.

#### 5.18.3.20 `begin()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 315 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::crend()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rend()`.

#### 5.18.3.21 `begin()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 326 of file `vstring.h`.

#### 5.18.3.22 `c_str()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::c_str ( ) const [inline],
[noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1647 of file `vstring.h`.

#### 5.18.3.23 `capacity()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity ( ) const [inline],
[noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 486 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

#### 5.18.3.24 cbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin ( ) const
[inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 390 of file vstring.h.

#### 5.18.3.25 cend()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 398 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

#### 5.18.3.26 clear()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear ( ) [inline], [noexcept]
```

Erases the string, making it empty.

Definition at line 514 of file vstring.h.

#### 5.18.3.27 compare() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) const [inline]
```

Compare to a string.

## Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2073 of file `vstring.h`.

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

5.18.3.28 `compare()` [2/6]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos,
    size_type __n,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) const
```

Compare substring to a string.

## Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 460 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__↵`  
`versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

#### 5.18.3.29 `compare()` [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos1,
    size_type __n1,
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos2,
    size_type __n2 ) const
```

Compare substring to a substring.

##### Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of str.
<code>__n2</code>	Number of characters in substring of str.

##### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 477 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `std::min()`.

#### 5.18.3.30 `compare()` [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (
    const _CharT * __s ) const
```

Compare to a C string.

## Parameters

<code>__s</code>	C string to compare against.
------------------	------------------------------

## Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 496 of file `vstring.tcc`.

5.18.3.31 `compare()` [5/6]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s ) const
```

Compare substring to a C string.

## Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 512 of file `vstring.tcc`.



**5.18.3.32 compare()** [6/6]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2 ) const
```

Compare substring against a character array.

**Parameters**

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of s.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `r1en` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(), __s, r1en)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `l0` has no special meaning.

Definition at line 529 of file `vstring.tcc`.

**5.18.3.33 copy()**

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::copy (
    _CharT * __s,
    size_type __n,
    size_type __pos = 0 ) const
```

Copy substring into C string.

## Parameters

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

## Returns

Number of characters actually copied

## Exceptions

<code>std::out_of_range</code>	If <code>pos &gt; size()</code> .
--------------------------------	-----------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 255 of file `vstring.tcc`.

5.18.3.34 `crbegin()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ( )
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 407 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

5.18.3.35 `crend()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ( )
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 416 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`.

#### 5.18.3.36 data()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data ( ) const [inline],
[noexcept]
```

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1657 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_↵not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `std::operator<<()`.

#### 5.18.3.37 empty()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty ( ) const [inline],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 522 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

#### 5.18.3.38 end() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 334 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin()`, and `__gnu_cxx::__versa_↵string< _CharT, _Traits, _Alloc, _Base >::rbegin()`.

**5.18.3.39** `end()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 345 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.18.3.40** `erase()` [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
    size_type __pos = 0,
    size_type __n = npos ) [inline]
```

Remove characters.

**Parameters**

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
--------------------------------	---

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1173 of file `vstring.h`.

**5.18.3.41** `erase()` [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
    const_iterator __position ) [inline]
```

Remove one character.

#### Parameters

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

#### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1190 of file `vstring.h`.

#### 5.18.3.42 `erase()` [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Remove a range of characters.

#### Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

#### Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1215 of file `vstring.h`.

5.18.3.43 `find()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const
```

Find position of a C substring.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>__s</code> to search for.

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 270 of file `vstring.tcc`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

5.18.3.44 `find()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a string.

## Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1693 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.18.3.45 find()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
    const _CharT * __s,
    size_type __pos = 0 ) const [inline]
```

Find position of a C string.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1708 of file `vstring.h`.

**5.18.3.46 find()** [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
    _CharT __c,
    size_type __pos = 0 ) const [noexcept]
```

Find position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 294 of file `vstring.tcc`.

5.18.3.47 `find_first_not_of()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1925 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.



5.18.3.48 `find_first_not_of()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const
```

Find position of a character not in C substring.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from s to consider.

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 392 of file `vstring.tcc`.

5.18.3.49 `find_first_not_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0 ) const [inline]
```

Find position of a character not in C string.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1956 of file `vstring.h`.

#### 5.18.3.50 `find_first_not_of()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
    _CharT __c,
    size_type __pos = 0 ) const [noexcept]
```

Find position of a different character.

##### Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 405 of file `vstring.tcc`.

#### 5.18.3.51 `find_first_of()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character of string.

##### Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1798 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.18.3.52 find\_first\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const
```

Find position of a character of C substring.

**Parameters**

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 353 of file `vstring.tcc`.

**5.18.3.53 find\_first\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
    const _CharT * __s,
    size_type __pos = 0 ) const [inline]
```

Find position of a character of C string.

## Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1828 of file `vstring.h`.

5.18.3.54 `find_first_of()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
    _CharT __c,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character.

## Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1847 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`.

5.18.3.55 `find_last_not_of()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1988 of file `vstring.h`.

5.18.3.56 `find_last_not_of()` [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const
```

Find last position of a character not in C substring.

## Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 417 of file `vstring.tcc`.

#### 5.18.3.57 `find_last_not_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos ) const [inline]
```

Find last position of a character not in C string.

##### Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

##### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2019 of file `vstring.h`.

#### 5.18.3.58 `find_last_not_of()` [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_not_of (
    _CharT __c,
    size_type __pos = npos ) const [noexcept]
```

Find last position of a different character.

##### Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 439 of file `vstring.tcc`.

**5.18.3.59 find\_last\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1862 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**5.18.3.60 find\_last\_of()** [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const
```

Find last position of a character of C substring.

## Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 370 of file `vstring.tcc`.

5.18.3.61 `find_last_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of (
    const _CharT * __s,
    size_type __pos = npos ) const [inline]
```

Find last position of a character of C string.

## Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1892 of file `vstring.h`.

5.18.3.62 `find_last_of()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of (
    _CharT __c,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character.



**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1911 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

**5.18.3.63 front()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::front ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 616 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`.

**5.18.3.64 front()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::front ( ) const
[inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 624 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`.

5.18.3.65 `get_allocator()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ( )
const [inline], [noexcept]
```

Return copy of allocator used to construct this string.

Definition at line 1664 of file `vstring.h`.

5.18.3.66 `insert()` [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    size_type __n,
    _CharT __c ) [inline]
```

Insert multiple characters.

## Parameters

<code>__p</code>	Const_iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

## Returns

Iterator referencing the first inserted char.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 940 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`.

**5.18.3.67 insert()** [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    _InputIterator __beg,
    _InputIterator __end ) [inline]
```

Insert a range of characters.

**Parameters**

<code>__p</code>	Const_iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 984 of file `vstring.h`.

**5.18.3.68 insert()** [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    std::initializer\_list< _CharT > __l ) [inline]
```

Insert an `initializer_list` of characters.

**Parameters**

<code>__p</code>	Const_iterator referencing location in string to insert at.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 1020 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`.

**5.18.3.69 `insert()`** [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos1,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Insert value of a string.

**Parameters**

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1037 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

## 5.18.3.70 insert() [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos1,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos2,
    size_type __n ) [inline]
```

Insert a substring.

## Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in str to insert.
<code>__n</code>	Number of characters to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> .

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1060 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

## 5.18.3.71 insert() [6/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos,
    const _CharT * __s,
    size_type __n ) [inline]
```

Insert a C substring.

## Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.
<code>__n</code>	The number of characters to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1083 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

5.18.3.72 `insert()` [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos,
    const _CharT * __s ) [inline]
```

Insert a C string.

## Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1102 of file `vstring.h`.

#### 5.18.3.73 `insert()` [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    size_type __pos,
    size_type __n,
    _CharT __c ) [inline]
```

Insert multiple characters.

##### Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

##### Returns

Reference to this string.

##### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1126 of file `vstring.h`.

#### 5.18.3.74 `insert()` [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
    const_iterator __p,
    _CharT __c ) [inline]
```

Insert one character.

## Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

## Returns

Iterator referencing newly inserted char.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1145 of file `vstring.h`.

5.18.3.75 `length()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length ( ) const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 431 of file `vstring.h`.

5.18.3.76 `max_size()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size ( ) const [inline],
[noexcept]
```

Returns the `size()` of the largest possible string.

Definition at line 436 of file `vstring.h`.

Referenced by `std::getline()`.



**5.18.3.77 operator+=()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Append a string to this string.

## Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

## Returns

Reference to this string.

Definition at line 651 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

**5.18.3.78** `operator+=()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
    const _CharT * __s ) [inline]
```

Append a C string.

## Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

## Returns

Reference to this string.

Definition at line 660 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

**5.18.3.79** `operator+=()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
    _CharT __c ) [inline]
```

Append a character.

## Parameters

<code>_↵</code>	The character to append.
<code>_C</code>	

## Returns

Reference to this string.

Definition at line 669 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::push_back()`.

5.18.3.80 `operator+=()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
    std::initializer_list< _CharT > __l ) [inline]
```

Append an `initializer_list` of characters.

## Parameters

<code>↵</code>	The <code>initializer_list</code> of characters to be appended.
<code>_↵</code>	
<code>↵</code>	
<code>_↵</code>	
<code>l</code>	

## Returns

Reference to this string.

Definition at line 682 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

5.18.3.81 `operator=()` [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Assign the value of `str` to this string.

## Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 256 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

5.18.3.82 `operator=()` [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
    __versa_string<_CharT, _Traits, _Alloc, _Base > && __str ) [inline], [noexcept]
```

String move assignment operator.

## Parameters

<code>__str</code>	Source string.
--------------------	----------------

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 268 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

5.18.3.83 `operator=()` [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
    std::initializer_list<_CharT > __l ) [inline]
```

Set value to string constructed from initializer list.

## Parameters

<code>↵</code>	<code>std::initializer_list.</code>
<code>↵</code>	
<code>↵</code>	
<code>↵</code>	
<code>/</code>	

Definition at line 280 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

#### 5.18.3.84 `operator=()` [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
    const _CharT * __s ) [inline]
```

Copy contents of `__s` into this string.

##### Parameters

<code>__s</code>	Source null-terminated string.
------------------	--------------------------------

Definition at line 292 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

#### 5.18.3.85 `operator=()` [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
    _CharT __c ) [inline]
```

Set value to string of length 1.

##### Parameters

<code>__c</code>	Source character.
------------------	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 303 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

**5.18.3.86** `operator[]()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
    size_type __pos ) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 537 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::front()`.

**5.18.3.87** `operator[]()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
    size_type __pos ) [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 554 of file `vstring.h`.

#### 5.18.3.88 pop\_back()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back ( ) [inline]
```

Remove the last character.

The string must be non-empty.

Definition at line 1235 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

#### 5.18.3.89 push\_back()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back (
    _CharT __c ) [inline]
```

Append a single character.

##### Parameters

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 788 of file vstring.h.

Referenced by `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=(())`.

#### 5.18.3.90 rbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ( ) [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 354 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

**5.18.3.91** `rbegin()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ( )
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 363 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

**5.18.3.92** `rend()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ( ) [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 372 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`.

**5.18.3.93** `rend()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ( )
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 381 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`.

**5.18.3.94** `replace()` [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos,
    size_type __n,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Replace characters with value from another string.



**Parameters**

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1257 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**5.18.3.95 replace()** [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos1,
    size_type __n1,
    const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos2,
    size_type __n2 ) [inline]
```

Replace characters with value from another string.

**Parameters**

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of <code>str</code> to use.
<code>__n2</code>	Number of characters from <code>str</code> to use.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1280 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**5.18.3.96 `replace()`** [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2 ) [inline]
```

Replace characters with value of a C substring.

**Parameters**

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.
<code>__n2</code>	Number of characters from <code>__s</code> to use.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1308 of file `vstring.h`.

#### 5.18.3.97 `replace()` [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s ) [inline]
```

Replace characters with value of a C string.

##### Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

##### Returns

Reference to this string.

##### Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1332 of file `vstring.h`.

#### 5.18.3.98 `replace()` [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    size_type __pos,
```

```

size_type __n1,
size_type __n2,
_CharT __c ) [inline]

```

Replace characters with multiple characters.

#### Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

#### Returns

Reference to this string.

#### Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1356 of file `vstring.h`.

#### 5.18.3.99 `replace()` [6/11]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str ) [inline]

```

Replace range of characters with string.

#### Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1375 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.18.3.100 replace()** [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    const _CharT * __s,
    size_type __n ) [inline]
```

Replace range of characters with C substring.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range  $[i1, i2)$ . In place, the first  $n$  characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1398 of file `vstring.h`.

#### 5.18.3.101 `replace()` [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    const _CharT * __s ) [inline]
```

Replace range of characters with C string.

##### Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

##### Returns

Reference to this string.

##### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range  $[i1, i2)$ . In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1424 of file `vstring.h`.





#### 5.18.3.102 `replace()` [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
```

```
const_iterator __i2,
size_type __n,
_CharT __c ) [inline]
```

Replace range of characters with multiple characters.

#### Parameters

<a href="#"></a> <code>__i1</code>	Iterator referencing start of range to replace.
<a href="#"></a> <code>__i2</code>	Iterator referencing end of range to replace.
<a href="#"></a> <code>__n</code>	Number of characters to insert.
<a href="#"></a> <code>__c</code>	Character to insert.

#### Returns

Reference to this string.

#### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1449 of file `vstring.h`.

#### 5.18.3.103 `replace()` [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2 ) [inline]
```

Replace range of characters with range.

#### Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1478 of file `vstring.h`.

**5.18.3.104 `replace()`** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1,
    const_iterator __i2,
    std::initializer_list<_CharT > __l ) [inline]
```

Replace range of characters with `initializer_list`.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1582 of file `vstring.h`.



References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

#### 5.18.3.105 `reserve()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::reserve (
    size_type __res_arg = 0 ) [inline]
```

Attempt to preallocate enough memory for specified number of characters.

##### Parameters

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

##### Exceptions

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 507 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

#### 5.18.3.106 `resize()` [1/2]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize (
    size_type __n,
    _CharT __c )
```

Resizes the string to the specified number of characters.

##### Parameters

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 50 of file `vstring.tcc`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize()`.

#### 5.18.3.107 `resize()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize (
    size_type __n ) [inline]
```

Resizes the string to the specified number of characters.

##### Parameters

<code>__n</code>	Number of characters the string should contain.
------------------	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 463 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize()`.

#### 5.18.3.108 `rfind()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind (
    const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a string.

##### Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1738 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

**5.18.3.109 rfind()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const
```

Find last position of a C substring.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 313 of file `vstring.tcc`.

**5.18.3.110 rfind()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
    const _CharT * __s,
    size_type __pos = npos ) const [inline]
```

Find last position of a C string.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1768 of file `vstring.h`.

**5.18.3.111 `rfind()`** [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
    _CharT __c,
    size_type __pos = npos ) const    [noexcept]
```

Find last position of a character.

**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 335 of file `vstring.tcc`.

**5.18.3.112 `shrink_to_fit()`**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit ( ) [inline],
[noexcept]
```

A non-binding request to reduce capacity() to size().

Definition at line 469 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

#### 5.18.3.113 size()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size ( ) const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 425 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`, `__gnu_cxx::operator+()`, `std::operator<<()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit()`.

#### 5.18.3.114 substr()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr (
    size_type __pos = 0,
    size_type __n = npos ) const [inline]
```

Get a substring.

Parameters

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

**Returns**

The new string.

**Exceptions**

<code>std::out_of_range</code>	If <code>pos &gt; size()</code> .
--------------------------------	-----------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2052 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string()`.

**5.18.3.115 swap()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap (
    __versa_string<_CharT, _Traits, _Alloc, _Base> & __s ) [inline], [noexcept]
```

Swap contents with another string.

**Parameters**

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1636 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`, and `__gnu_cxx::swap()`.

**5.18.4 Member Data Documentation****5.18.4.1 npos**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_
CharT, _Traits, _Alloc, _Base>::npos [static]
```

Value returned by various member functions when they fail.

Definition at line 81 of file `vstring.h`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

## 5.19 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

### Public Types

- `typedef _ToType::element_type * type`

### 5.19.1 Detailed Description

```
template<typename _ToType>
struct __gnu_cxx::_Caster<_ToType>
```

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file `cast.h`.

The documentation for this struct was generated from the following file:

- [cast.h](#)

## 5.20 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

### Public Types

- `typedef unsigned long int_type`
- `typedef std::streamoff off_type`
- `typedef std::streampos pos_type`
- `typedef std::mbstate\_t state_type`

## 5.20.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::_Char_types<_CharT>
```

Mapping from character type to associated types.

## Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, streamoff, streampos, and mbstate\_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 62 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

5.21 `__gnu_cxx::_ExtPtr_allocator<_Tp>` Class Template Reference

## Public Types

- typedef `_Pointer_adapter<_Relative_pointer_impl<const _Tp>>` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Pointer_adapter<_Relative_pointer_impl<_Tp>>` **pointer**
- typedef `_Tp &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **\_ExtPtr\_allocator** (const `_ExtPtr_allocator` &\_\_rarg) noexcept
- template<typename \_Up >  
  **\_ExtPtr\_allocator** (const `_ExtPtr_allocator<_Up>` &\_\_rarg) noexcept
- const `std::allocator<_Tp>` & **\_M\_getUnderlyingImp** () const
- **pointer address** (reference \_\_x) const noexcept
- **const\_pointer address** (const\_reference \_\_x) const noexcept
- **pointer allocate** (size\_type \_\_n, void \* \_\_hint=0)
- template<typename \_Up, typename... \_Args>  
  void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- template<typename... \_Args>  
  void **construct** (pointer \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up >  
  void **destroy** (\_Up \* \_\_p)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const noexcept
- template<typename \_Up >  
  bool **operator!=** (const `_ExtPtr_allocator<_Up>` &\_\_rarg)
- bool **operator!=** (const `_ExtPtr_allocator` &\_\_rarg)
- template<typename \_Up >  
  bool **operator==** (const `_ExtPtr_allocator<_Up>` &\_\_rarg)
- bool **operator==** (const `_ExtPtr_allocator` &\_\_rarg)



## Friends

- `template<typename _Up >`  
`void swap (\_ExtPtr\_allocator< _Up > &, \_ExtPtr\_allocator< _Up > &)`

### 5.21.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator< _Tp >
```

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.

Definition at line 56 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

## 5.22 \_\_gnu\_cxx::\_Invalid\_type Struct Reference

### 5.22.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

Definition at line 213 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.23 \_\_gnu\_cxx::\_Pointer\_adapter< \_Storage\_policy > Class Template Reference

Inherits `_Storage_policy`.

### Public Types

- `typedef std::ptrdiff_t difference_type`
- `typedef _Storage_policy::element_type element_type`
- `typedef std::random\_access\_iterator\_tag iterator_category`
- `typedef \_Pointer\_adapter pointer`
- `typedef _Reference_type< element_type >::reference reference`
- `typedef \_Unqualified\_type< element_type >::type value_type`

## Public Member Functions

- `__Pointer_adapter` (`element_type * __arg=0`)
- `__Pointer_adapter` (`const __Pointer_adapter & __arg`)
- `template<typename _Up >`  
`__Pointer_adapter` (`_Up * __arg`)
- `template<typename _Up >`  
`__Pointer_adapter` (`const __Pointer_adapter< _Up > & __arg`)
- `operator __unspecified_bool_type` () `const`
- `bool operator!` () `const`
- `reference operator*` () `const`
- `__Pointer_adapter & operator++` ()
- `__Pointer_adapter operator++` (`int`)
- `__Pointer_adapter & operator+=` (`short __offset`)
- `__Pointer_adapter & operator+=` (`unsigned short __offset`)
- `__Pointer_adapter & operator+=` (`int __offset`)
- `__Pointer_adapter & operator+=` (`unsigned int __offset`)
- `__Pointer_adapter & operator+=` (`long __offset`)
- `__Pointer_adapter & operator+=` (`unsigned long __offset`)
- `__Pointer_adapter & operator+=` (`long long __offset`)
- `__Pointer_adapter & operator+=` (`unsigned long long __offset`)
- `template<typename _Up >`  
`std::ptrdiff_t operator-` (`const __Pointer_adapter< _Up > & __rhs`) `const`
- `__Pointer_adapter & operator--` ()
- `__Pointer_adapter operator--` (`int`)
- `__Pointer_adapter & operator-=` (`short __offset`)
- `__Pointer_adapter & operator-=` (`unsigned short __offset`)
- `__Pointer_adapter & operator-=` (`int __offset`)
- `__Pointer_adapter & operator-=` (`unsigned int __offset`)
- `__Pointer_adapter & operator-=` (`long __offset`)
- `__Pointer_adapter & operator-=` (`unsigned long __offset`)
- `__Pointer_adapter & operator-=` (`long long __offset`)
- `__Pointer_adapter & operator-=` (`unsigned long long __offset`)
- `element_type * operator->` () `const`
- `__Pointer_adapter & operator=` (`const __Pointer_adapter & __arg`)
- `template<typename _Up >`  
`__Pointer_adapter & operator=` (`const __Pointer_adapter< _Up > & __arg`)
- `template<typename _Up >`  
`__Pointer_adapter & operator=` (`_Up * __arg`)
- `reference operator[]` (`std::ptrdiff_t __index`) `const`

## Friends

- `__Pointer_adapter operator+` (`const __Pointer_adapter & __lhs`, `short __offset`)
- `__Pointer_adapter operator+` (`short __offset`, `const __Pointer_adapter & __rhs`)
- `__Pointer_adapter operator+` (`const __Pointer_adapter & __lhs`, `unsigned short __offset`)
- `__Pointer_adapter operator+` (`unsigned short __offset`, `const __Pointer_adapter & __rhs`)
- `__Pointer_adapter operator+` (`const __Pointer_adapter & __lhs`, `int __offset`)
- `__Pointer_adapter operator+` (`int __offset`, `const __Pointer_adapter & __rhs`)
- `__Pointer_adapter operator+` (`const __Pointer_adapter & __lhs`, `unsigned int __offset`)

- [\\_Pointer\\_adapter](#) **operator+** (unsigned int \_\_offset, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) & \_\_lhs, long \_\_offset)
- [\\_Pointer\\_adapter](#) **operator+** (long \_\_offset, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) & \_\_lhs, unsigned long \_\_offset)
- [\\_Pointer\\_adapter](#) **operator+** (unsigned long \_\_offset, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) & \_\_lhs, long long \_\_offset)
- [\\_Pointer\\_adapter](#) **operator+** (long long \_\_offset, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (unsigned long long \_\_offset, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [\\_Pointer\\_adapter](#) **operator+** (const [\\_Pointer\\_adapter](#) & \_\_lhs, unsigned long long \_\_offset)
- [std::ptrdiff\\_t](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, element\_type \* \_\_rhs)
- [std::ptrdiff\\_t](#) **operator-** (element\_type \* \_\_lhs, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [template<typename \\_Up >](#)  
[std::ptrdiff\\_t](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, \_Up \* \_\_rhs)
- [template<typename \\_Up >](#)  
[std::ptrdiff\\_t](#) **operator-** (\_Up \* \_\_lhs, const [\\_Pointer\\_adapter](#) & \_\_rhs)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, short \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, unsigned short \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, int \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, unsigned int \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, long \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, unsigned long \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, long long \_\_offset)
- [\\_Pointer\\_adapter](#) **operator-** (const [\\_Pointer\\_adapter](#) & \_\_lhs, unsigned long long \_\_offset)

### 5.23.1 Detailed Description

```
template<typename _Storage_policy>
class __gnu_cxx::Pointer_adapter< _Storage_policy >
```

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp> >; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const _Pointer_
adapter<_Std_pointer_impl<const _Tp> >;
```

Definition at line 281 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.24 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference

## Public Types

- `typedef _Tp element_type`

## Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (_Tp *__arg)`

## 5.24.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 109 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.25 `__gnu_cxx::_Relative_pointer_impl<const _Tp>` Class Template Reference

## Public Types

- `typedef const _Tp element_type`

## Public Member Functions

- `const _Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (const _Tp *__arg)`

### 5.25.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::Relative_pointer_impl< const _Tp >
```

Relative\_pointer\_impl needs a specialization for const T because of the casting done during pointer arithmetic.

Definition at line 161 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.26 \_\_gnu\_cxx::Std\_pointer\_impl< \_Tp > Class Template Reference

### Public Types

- typedef \_Tp **element\_type**

### Public Member Functions

- \_Tp \* **get** () const
- bool **operator**< (const [\\_Std\\_pointer\\_impl](#) &\_\_rarg) const
- bool **operator**== (const [\\_Std\\_pointer\\_impl](#) &\_\_rarg) const
- void **set** (element\_type \*\_\_arg)

### 5.26.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::Std_pointer_impl< _Tp >
```

A storage policy for use with [\\_Pointer\\_adapter](#)<> which yields a standard pointer.

A [\\_Storage\\_policy](#) is required to provide 4 things: 1) A get() API for returning the stored pointer value. 2) An set() API for storing a pointer value. 3) An element\_type typedef to define the type this points to. 4) An operator<() to support pointer comparison. 5) An operator==() to support pointer comparison.

Definition at line 66 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.27 `__gnu_cxx::_Unqualified_type<_Tp>` Struct Template Reference

## Public Types

- `typedef _Tp type`

## 5.27.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::_Unqualified_type<_Tp>
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

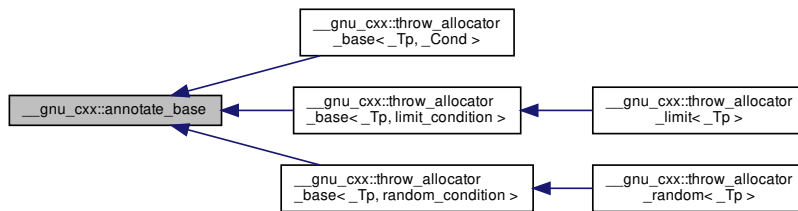
Definition at line 241 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

5.28 `__gnu_cxx::annotate_base` Struct Reference

Inheritance diagram for `__gnu_cxx::annotate_base`:



## Public Member Functions

- void **check** (size\_t label)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_constructed** (void \*p)
- void **check\_constructed** (size\_t label)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)

### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

### Friends

- [std::ostream](#) & **operator**<< ([std::ostream](#) &, const [annotate\\_base](#) &)

#### 5.28.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

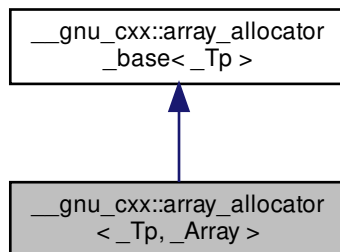
Definition at line 88 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.29 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference

Inheritance diagram for `__gnu_cxx::array_allocator<_Tp, _Array>`:



### Public Types

- typedef `_Array` **array\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `std::true_type` **is\_always\_equal**
- typedef `_Tp *` **pointer**
- typedef `std::true_type` **propagate\_on\_container\_move\_assignment**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **array\_allocator** (array\_type \*\_\_array=0) noexcept
- **array\_allocator** (const [array\\_allocator](#) &\_\_o) noexcept
- template<typename \_Tp1 , typename \_Array1 >  
**array\_allocator** (const [array\\_allocator](#)< \_Tp1, \_Array1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*=0)
- template<typename \_Up , typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer, size\_type)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

## 5.29.1 Detailed Description

```
template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>>
class __gnu_cxx::array_allocator<_Tp, _Array>
```

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

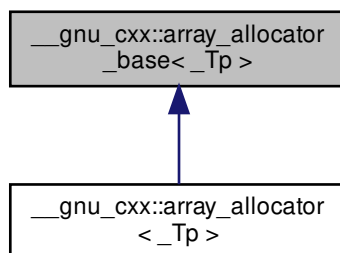
Definition at line 110 of file `array_allocator.h`.

The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

5.30 `__gnu_cxx::array_allocator_base<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::array_allocator_base<_Tp>`:





## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer, size\_type)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

## 5.30.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::array_allocator_base< _Tp >
```

Base class.

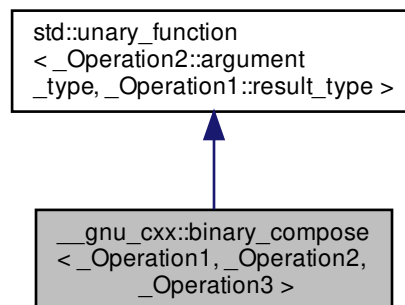
Definition at line 54 of file array\_allocator.h.

The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

## 5.31 \_\_gnu\_cxx::binary\_compose&lt; \_Operation1, \_Operation2, \_Operation3 &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- **`binary_compose`** (`const _Operation1 &__x`, `const _Operation2 &__y`, `const _Operation3 &__z`)
- `_Operation1::result_type` **`operator()`** (`const typename _Operation2::argument_type &__x`) `const`

### Protected Attributes

- `_Operation1` **`_M_fn1`**
- `_Operation2` **`_M_fn2`**
- `_Operation3` **`_M_fn3`**

#### 5.31.1 Detailed Description

```
template<class _Operation1, class _Operation2, class _Operation3>
class __gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3 >
```

An [SGI extension](#) .

Definition at line 150 of file `ext/functional`.

#### 5.31.2 Member Typedef Documentation

##### 5.31.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary\_function<\_Arg, \_Result >::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

### 5.31.2.2 result\_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary\_function< _Arg, _Result >::result\_type [inherited]
```

result\_type is the return type

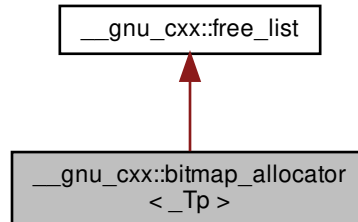
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.32 \_\_gnu\_cxx::bitmap\_allocator< \_Tp > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::bitmap\_allocator< \_Tp >:



### Public Types

- typedef free\_list::\_\_mutex\_type **\_\_mutex\_type**
- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **bitmap\_allocator** (const [bitmap\\_allocator](#) &) noexcept
- `template<typename _Tp1 >`  
**bitmap\_allocator** (const [bitmap\\_allocator](#)<\_Tp1> &) noexcept
- pointer [\\_M\\_allocate\\_single\\_object](#) ()
- void [\\_M\\_deallocate\\_single\\_object](#) (pointer \_\_p) throw ()
- pointer **address** (reference \_\_r) const noexcept
- const\_pointer **address** (const\_reference \_\_r) const noexcept
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, typename [bitmap\\_allocator](#)< void >::const\_pointer)
- `template<typename _Up, typename... _Args>`  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) throw ()
- `template<typename _Up >`  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

## Private Types

- typedef vector\_type::iterator **iterator**
- typedef [\\_\\_detail::\\_\\_mini\\_vector](#)< value\_type > **vector\_type**

## Private Member Functions

- void [\\_M\\_clear](#) ()
- size\_t \* [\\_M\\_get](#) (size\_t \_\_sz)
- void [\\_M\\_insert](#) (size\_t \*\_\_addr) throw ()

## 5.32.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::bitmap_allocator<_Tp>
```

Bitmap Allocator, primary template.

Definition at line 660 of file `bitmap_allocator.h`.

## 5.32.2 Member Function Documentation

5.32.2.1 `_M_allocate_single_object()`

```
template<typename _Tp >
pointer __gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object ( ) [inline]
```

Allocates memory for a single object of size `sizeof(_Tp)`.

## Exceptions

<code>std::bad_alloc.</code>	If memory can not be allocated.
------------------------------	---------------------------------

Complexity: Worst case complexity is  $O(N)$ , but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of  $O(1)$ ! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 824 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::Bit_scan←_forward()`.

### 5.32.2.2 `_M_deallocate_single_object()`

```
template<typename _Tp >
void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (
    pointer __p ) throw ( )    [inline]
```

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity:  $O(\lg(N))$ , but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in  $O(1)$  time by the `deallocate` function.

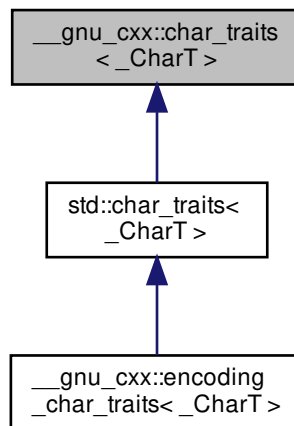
Definition at line 914 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.33 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



## Public Types

- `typedef _CharT char_type`
- `typedef _Char_types<_CharT>::int_type int_type`
- `typedef _Char_types<_CharT>::off_type off_type`
- `typedef _Char_types<_CharT>::pos_type pos_type`
- `typedef _Char_types<_CharT>::state_type state_type`

## Static Public Member Functions

- `static _GLIBCXX14_CONSTEXPR void assign (char_type &__c1, const char_type &__c2)`
- `static char_type * assign (char_type *__s, std::size_t __n, char_type __a)`
- `static _GLIBCXX14_CONSTEXPR int compare (const char_type *__s1, const char_type *__s2, std::size_t __n)`
- `static char_type * copy (char_type *__s1, const char_type *__s2, std::size_t __n)`
- `static constexpr int_type eof ()`
- `static constexpr bool eq (const char_type &__c1, const char_type &__c2)`
- `static constexpr bool eq_int_type (const int_type &__c1, const int_type &__c2)`
- `static _GLIBCXX14_CONSTEXPR const char_type * find (const char_type *__s, std::size_t __n, const char_type &__a)`
- `static _GLIBCXX14_CONSTEXPR std::size_t length (const char_type *__s)`
- `static constexpr bool lt (const char_type &__c1, const char_type &__c2)`
- `static char_type * move (char_type *__s1, const char_type *__s2, std::size_t __n)`
- `static constexpr int_type not_eof (const int_type &__c)`
- `static constexpr char_type to_char_type (const int_type &__c)`
- `static constexpr int_type to_int_type (const char_type &__c)`

## 5.33.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::char_traits<_CharT>
```

Base class used to implement `std::char_traits`.

## Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 87 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

### 5.34 `__gnu_cxx::character< _Value, _Int, _St >` Struct Template Reference

#### Public Types

- typedef `character< _Value, _Int, _St >` **char\_type**
- typedef `_Int` **int\_type**
- typedef `_St` **state\_type**
- typedef `_Value` **value\_type**

#### Static Public Member Functions

- template<typename V2 >  
static `char_type` **from** (const V2 &v)
- template<typename V2 >  
static V2 **to** (const `char_type` &c)

#### Public Attributes

- `value_type` **value**

#### 5.34.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character< _Value, _Int, _St >
```

A POD class that serves as a character abstraction class.

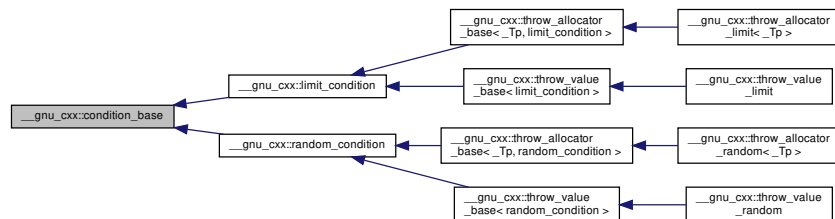
Definition at line 49 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

### 5.35 `__gnu_cxx::condition_base` Struct Reference

Inheritance diagram for `__gnu_cxx::condition_base`:



## 5.35.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature `void throw_conditionally()`

Definition at line 403 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.36 `__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>` Struct Template Reference

Inherits `__gnu_cxx::Constant_binary_fun<_Result, _Arg1, _Arg2>`.

## Public Types

- `typedef _Arg1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Arg2 second_argument_type`

## Public Member Functions

- `constant_binary_fun` (`const _Result &__v`)
- `const result_type &operator()` (`const _Arg1 &`, `const _Arg2 &`) `const`

## Public Attributes

- `_Result _M_val`

## 5.36.1 Detailed Description

```
template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>
struct __gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>
```

An [SGI extension](#) .

Definition at line 320 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)



### 5.37 `__gnu_cxx::constant_unary_fun<_Result, _Argument >` Struct Template Reference

Inherits `__gnu_cxx::Constant_unary_fun<_Result, _Argument >`.

#### Public Types

- typedef `_Argument` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- **constant\_unary\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** (const `_Argument` &) const

#### Public Attributes

- `result_type` **\_M\_val**

#### 5.37.1 Detailed Description

```
template<class _Result, class _Argument = _Result>
struct __gnu_cxx::constant_unary_fun<_Result, _Argument >
```

An [SGI extension](#) .

Definition at line 312 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 5.38 `__gnu_cxx::constant_void_fun<_Result >` Struct Template Reference

Inherits `__gnu_cxx::Constant_void_fun<_Result >`.

#### Public Types

- typedef `_Result` **result\_type**

#### Public Member Functions

- **constant\_void\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** () const

## Public Attributes

- `result_type` **`_M_val`**

## 5.38.1 Detailed Description

```
template<class _Result>
struct __gnu_cxx::constant_void_fun<_Result>
```

An [SGI extension](#) .

Definition at line 303 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.39 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

## Public Types

- `typedef _Traits::const_pointer` **`const_pointer`**
- `typedef _Traits::const_reference` **`const_reference`**
- `typedef _Traits::difference_type` **`difference_type`**
- `typedef _Traits::pointer` **`pointer`**
- `typedef _Traits::reference` **`reference`**
- `typedef _Traits::size_type` **`size_type`**
- `typedef _Traits::value_type` **`value_type`**

## Public Member Functions

- `template<typename _Alloc2>`  
**`debug_allocator`** (const [debug\\_allocator](#)<\_Alloc2> &\_\_a2, typename `__convertible<_Alloc2>::__type`=0)
- **`debug_allocator`** (const `_Alloc` &\_\_a)
- `pointer` **`allocate`** (size\_type \_\_n)
- `pointer` **`allocate`** (size\_type \_\_n, const void \* \_\_hint)
- void **`construct`** (pointer \_\_p, const value\_type &\_\_val)
- `template<typename _Tp, typename... _Args>`  
void **`construct`** (\_Tp \* \_\_p, \_Args &&... \_\_args)
- void **`deallocate`** (pointer \_\_p, size\_type \_\_n)
- `template<typename _Tp>`  
void **`destroy`** (\_Tp \* \_\_p)
- size\_type **`max_size`** () const throw ()

## Friends

- `template<typename >`  
class **debug\_allocator**
- `bool operator== (const debug\_allocator &__lhs, const debug\_allocator &__rhs)`

## 5.39.1 Detailed Description

```
template<typename _Alloc>
class __gnu_cxx::debug_allocator< _Alloc >
```

A meta-allocator with debugging bits.

This is precisely the allocator defined in the C++03 Standard.

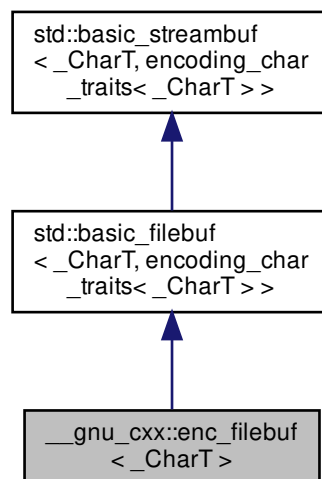
Definition at line 62 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

5.40 `__gnu_cxx::enc_filebuf< _CharT >` Class Template Reference

Inheritance diagram for `__gnu_cxx::enc_filebuf< _CharT >`:



## Public Types

- `typedef codecvt< char\_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic\_filebuf< char\_type, traits\_type > __filebuf_type`
- `typedef traits_type::state_type __state_type`
- `typedef basic\_streambuf< char\_type, traits\_type > __streambuf_type`
- `typedef _CharT char\_type`
- `typedef traits_type::int_type int\_type`
- `typedef traits_type::off_type off\_type`
- `typedef traits_type::pos_type pos\_type`
- `typedef traits_type::state_type state\_type`
- `typedef encoding\_char\_traits< _CharT > traits\_type`

## Public Member Functions

- `enc\_filebuf (state_type &__state)`
  - `\_\_filebuf\_type * close ()`
  - `locale getloc () const`
  - `streamsize in\_avail ()`
  - `bool is\_open () const throw ()`
  - `\_\_filebuf\_type * open (const char *__s, ios_base::openmode __mode)`
  - `\_\_filebuf\_type * open (const std::string &__s, ios_base::openmode __mode)`
  - `locale pubimbue (const locale &__loc)`
  - `int\_type sbumpc ()`
  - `int\_type sgetc ()`
  - `streamsize sgetn (char\_type *__s, streamsize __n)`
  - `int\_type snextc ()`
  - `int\_type sputbackc (char\_type __c)`
  - `int\_type sputc (char\_type __c)`
  - `streamsize sputn (const char\_type *__s, streamsize __n)`
  - `int\_type sungetc ()`
  - `void swap (basic\_filebuf &)`
- 
- `basic\_streambuf * pubsetbuf (char\_type *__s, streamsize __n)`
  - `pos\_type pubseekoff (off\_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `pos\_type pubseekpos (pos\_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `int pubsync ()`

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
  - void **\_\_safe\_pbump** (streamsize \_\_n)
  - void **\_M\_allocate\_internal\_buffer** ()
  - bool **\_M\_convert\_to\_external** (char\_type \*, streamsize)
  - void **\_M\_create\_pback** ()
  - void **\_M\_destroy\_internal\_buffer** () throw ()
  - void **\_M\_destroy\_pback** () throw ()
  - int **\_M\_get\_ext\_pos** (\_\_state\_type &\_\_state)
  - pos\_type **\_M\_seek** (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
  - void **\_M\_set\_buffer** (streamsize \_\_off)
  - bool **\_M\_terminate\_output** ()
  - void **gbump** (int \_\_n)
  - virtual void **imbue** (const locale &\_\_loc)
  - virtual void **imbue** (const locale &\_\_loc \_\_attribute\_\_((\_\_unused\_\_)))
  - virtual int\_type **overflow** (int\_type \_\_c=encoding\_char\_traits<\_CharT>::eof())
  - virtual int\_type **overflow** (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
  - virtual int\_type **pbackfail** (int\_type \_\_c=encoding\_char\_traits<\_CharT>::eof())
  - virtual int\_type **pbackfail** (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
  - void **pbump** (int \_\_n)
  - virtual pos\_type **seekoff** (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual pos\_type **seekpos** (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual \_\_streambuf\_type \* **setbuf** (char\_type \*\_\_s, streamsize \_\_n)
  - void **setg** (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
  - void **setp** (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
  - virtual streamsize **showmanyc** ()
  - void **swap** (basic\_streambuf &\_\_sb)
  - virtual int **sync** ()
  - virtual int\_type **uflow** ()
  - virtual int\_type **underflow** ()
  - virtual streamsize **xsggetn** (char\_type \*\_\_s, streamsize \_\_n)
  - virtual streamsize **xspu** (const char\_type \*\_\_s, streamsize \_\_n)
- 
- char\_type \* **eback** () const
  - char\_type \* **gptr** () const
  - char\_type \* **egptr** () const
- 
- char\_type \* **pbase** () const
  - char\_type \* **pptr** () const
  - char\_type \* **epptr** () const

## Protected Attributes

- `char_type * _M_buf`
  - `bool _M_buf_allocated`
  - `locale _M_buf_locale`
  - `size_t _M_buf_size`
  - `const __codecvt_type * _M_codecvt`
  - `char * _M_ext_buf`
  - `streamsize _M_ext_buf_size`
  - `char * _M_ext_end`
  - `const char * _M_ext_next`
  - `__file_type _M_file`
  - `char_type * _M_in_beg`
  - `char_type * _M_in_cur`
  - `char_type * _M_in_end`
  - `__c_lock _M_lock`
  - `ios_base::openmode _M_mode`
  - `char_type * _M_out_beg`
  - `char_type * _M_out_cur`
  - `char_type * _M_out_end`
  - `bool _M_reading`
  - `__state_type _M_state_beg`
  - `__state_type _M_state_cur`
  - `__state_type _M_state_last`
  - `bool _M_writing`
- 
- `char_type _M_pback`
  - `char_type * _M_pback_cur_save`
  - `char_type * _M_pback_end_save`
  - `bool _M_pback_init`

## 5.40.1 Detailed Description

```
template<typename _CharT>
class __gnu_cxx::enc_filebuf<_CharT>
```

class `enc_filebuf`.

Definition at line 42 of file `enc_filebuf.h`.

## 5.40.2 Member Function Documentation

#### 5.40.2.1 `_M_create_pback()`

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_create_pback ( ) [inline],  
[protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 199 of file `fstream`.

#### 5.40.2.2 `_M_destroy_pback()`

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_destroy_pback ( ) throw ( )  
[inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 216 of file `fstream`.

#### 5.40.2.3 `_M_set_buffer()`

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_set_buffer (   
    streamsize __off ) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 443 of file `fstream`.

#### 5.40.2.4 `close()`

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__filebuf_type * std::basic_filebuf< ↵  
_CharT, encoding_char_traits< _CharT > >::close ( ) [inherited]
```

Closes the currently associated file.

##### Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 213 of file `fstream.tcc`.

#### 5.40.2.5 `eback()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

#### 5.40.2.6 `egptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

#### 5.40.2.7 `epptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 542 of file `streambuf`.

#### 5.40.2.8 `gbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the read position.



**Parameters**

<code>_↔ _n</code>	The delta by which to move.
------------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

**5.40.2.9 getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

**5.40.2.10 gptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

**5.40.2.11 imbue()**

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale &__loc __attribute__((unused)) ) [inline], [protected], [virtual],
[inherited]
```

Changes translations.

## Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Definition at line 583 of file `streambuf`.

5.40.2.12 `in_avail()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail ( ) [inline], [inherited]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

5.40.2.13 `is_open()`

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::is_open ( ) const throw ( )
[inline], [inherited]
```

Returns true if the external file is open.

Definition at line 260 of file `fstream`.

5.40.2.14 `open()` [1/2]

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__filebuf_type * std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open (
    const char * __s,
    ios_base::openmode __mode ) [inherited]
```

Opens an external file.

## Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

## Returns

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

ios_base Flag combination					stdio equivalent
binary	in	out	trunc	app	
		+			w
		+		+	a
				+	a
		+	+		w
	+				r
	+	+			r+
	+	+	+		w+
	+	+		+	a+
	+			+	a+
+		+			wb
+		+		+	ab
+				+	ab
+		+	+		wb
+	+				rb
+	+	+			r+b
+	+	+	+		w+b
+	+	+		+	a+b
+	+			+	a+b

Definition at line 179 of file `fstream.tcc`.

5.40.2.15 `open()` [2/2]

```
__filebuf_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::open (
    const std::string & __s,
    ios_base::openmode __mode ) [inline], [inherited]
```

Opens an external file.

## Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

## Returns

`this` on success, NULL on failure

Definition at line 315 of file `fstream`.

5.40.2.16 `overflow()`

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::overflow (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

## Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

## Note

Base class version does nothing, returns `eof()`.

Definition at line 775 of file `streambuf`.

#### 5.40.2.17 pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
    int_type __c __attribute__((__unused__)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Tries to back up the input sequence.

## Parameters

<code>_↔</code>	The character to be inserted back into the sequence.
<code>_C</code>	

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Definition at line 731 of file `streambuf`.

5.40.2.18 `pbase()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

5.40.2.19 `pbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

<code>_↔</code>	The delta by which to move.
<code>_n</code>	

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**5.40.2.20 pptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**5.40.2.21 pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc ) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

5.40.2.22 `pubseekoff()`

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekoff` function.

Definition at line 258 of file `streambuf`.

5.40.2.23 `pubseekpos()`

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 270 of file `streambuf`.

5.40.2.24 `pubsetbuf()`

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```



Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 5.40.2.25 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]
```

Calls virtual sync function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 5.40.2.26 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 5.40.2.27 `seekoff()`

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 798 of file `fstream.tcc`.

5.40.2.28 `seekpos()`

```
basic_filebuf<_CharT, encoding_char_traits<_CharT> >::pos_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT> >::seekpos (
    pos_type __pos,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 858 of file `fstream.tcc`.

5.40.2.29 `setbuf()`

```
basic_filebuf<_CharT, encoding_char_traits<_CharT> >::__streambuf_type * std::basic_filebuf<
_CharT, encoding_char_traits<_CharT> >::setbuf (
    char_type * __s,
    streamsize __n ) [protected], [virtual], [inherited]
```

Manipulates the buffer.

**Parameters**

<code>__s</code>	Pointer to a buffer area.
<code>__n</code>	Size of <code>__s</code> .

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 769 of file `fstream.tcc`.

#### 5.40.2.30 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected], [inherited]
```

Setting the three read area pointers.

##### Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

##### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

#### 5.40.2.31 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend ) [inline], [protected], [inherited]
```

Setting the three write area pointers.

##### Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

##### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

5.40.2.32 `sgetc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

5.40.2.33 `sgetn()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

5.40.2.34 `showmanyc()`

```
streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::showmanyc ( ) [protected],
[virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 263 of file `fstream.tcc`.

**5.40.2.35 `snextc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**5.40.2.36 `sputbackc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
    char_type __c ) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

<code>__c</code>	The character to push back.
------------------	-----------------------------

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**5.40.2.37 `sputc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
    char_type __c ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

<code>__c</code>	A character to output.
------------------	------------------------

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.40.2.38 `sputn()`**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**5.40.2.39 `sungetc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**5.40.2.40 `sync()`**

```
int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync (
    void ) [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 978 of file `fstream.tcc`.

5.40.2.41 `uflow()`

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::uflow ( ) [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 707 of file `streambuf`.

5.40.2.42 `underflow()`

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type std::basic_filebuf<_CharT,  
encoding_char_traits<_CharT>>::underflow ( ) [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input `streambuf` can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 289 of file `fstream.tcc`.

5.40.2.43 `xsgetn()`

```
streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsgetn (  
    char_type * __s,  
    streamsize __n ) [protected], [virtual], [inherited]
```

Multiple character extraction.



**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 635 of file `fstream.tcc`.

**5.40.2.44 xspn()**

```
streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xspn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 721 of file `fstream.tcc`.

### 5.40.3 Member Data Documentation

#### 5.40.3.1 `_M_buf`

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf [protected],  
[inherited]
```

Pointer to the beginning of internal buffer.

Definition at line 136 of file `fstream`.

#### 5.40.3.2 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>  
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file `streambuf`.

#### 5.40.3.3 `_M_buf_size`

```
size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_size [protected],  
[inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 143 of file `fstream`.

#### 5.40.3.4 `_M_ext_buf`

```
char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf [protected],  
[inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 178 of file `fstream`.

#### 5.40.3.5 `_M_ext_buf_size`

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected],  
[inherited]
```

Size of buffer held by `_M_ext_buf`.

Definition at line 183 of file `fstream`.

#### 5.40.3.6 `_M_ext_next`

```
const char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected],  
[inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 190 of file `fstream`.

#### 5.40.3.7 `_M_in_beg`

```
template<typename _CharT, typename _Traits>  
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file `streambuf`.

#### 5.40.3.8 `_M_in_cur`

```
template<typename _CharT, typename _Traits>  
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file `streambuf`.

#### 5.40.3.9 `_M_in_end`

```
template<typename _CharT, typename _Traits>  
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file `streambuf`.

#### 5.40.3.10 `_M_mode`

```
ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_mode [protected],  
[inherited]
```

Place to stash in || out || in | out settings for current filebuf.

Definition at line 121 of file `fstream`.

#### 5.40.3.11 `_M_out_beg`

```
template<typename _CharT, typename _Traits>  
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file `streambuf`.

#### 5.40.3.12 `_M_out_cur`

```
template<typename _CharT, typename _Traits>  
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file `streambuf`.

#### 5.40.3.13 `_M_out_end`

```
template<typename _CharT, typename _Traits>  
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file `streambuf`.

#### 5.40.3.14 `_M_pback`

```
char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback [protected],  
[inherited]
```

Necessary bits for putback buffer management.

#### Note

pbacks of over one character are not currently supported.

Definition at line 164 of file `fstream`.

#### 5.40.3.15 `_M_pback_cur_save`

```
char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_cur_save [protected],  
[inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 165 of file fstream.

#### 5.40.3.16 `_M_pback_end_save`

```
char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save [protected],  
[inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 166 of file fstream.

#### 5.40.3.17 `_M_pback_init`

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init [protected],  
[inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 167 of file fstream.

## 5.40.3.18 \_M\_reading

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_reading [protected],
[inherited]
```

\_M\_reading == false && \_M\_writing == false for **uncommitted** mode; \_M\_reading == true for **read** mode; \_M\_writing == true for **write** mode;

NB: \_M\_reading == true && \_M\_writing == true is unused.

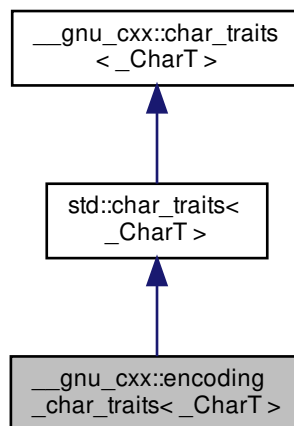
Definition at line 155 of file fstream.

The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

## 5.41 \_\_gnu\_cxx::encoding\_char\_traits&lt;\_CharT&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_cxx::encoding\_char\_traits<\_CharT>:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `std::fpos<state_type>` **pos\_type**
- typedef `encoding_state` **state\_type**

### Static Public Member Functions

- static \_GLIBCXX14\_CONSTEXPR void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static \_GLIBCXX14\_CONSTEXPR int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static \_GLIBCXX14\_CONSTEXPR const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static \_GLIBCXX14\_CONSTEXPR std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 5.41.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::encoding_char_traits< _CharT >
```

encoding\_char\_traits

Definition at line 211 of file codecvt\_specializations.h.

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.42 \_\_gnu\_cxx::encoding\_state Class Reference

### Public Types

- typedef iconv\_t **descriptor\_type**

### Public Member Functions

- **encoding\_state** (const char \* \_\_int, const char \* \_\_ext, int \_\_ibom=0, int \_\_ebom=0, int \_\_bytes=1)
- **encoding\_state** (const [encoding\\_state](#) & \_\_obj)
- int **character\_ratio** () const
- int **external\_bom** () const
- const [std::string](#) **external\_encoding** () const
- bool **good** () const throw ()
- const descriptor\_type & **in\_descriptor** () const
- int **internal\_bom** () const
- const [std::string](#) **internal\_encoding** () const
- [encoding\\_state](#) & **operator=** (const [encoding\\_state](#) & \_\_obj)
- const descriptor\_type & **out\_descriptor** () const

## Protected Member Functions

- void **construct** (const [encoding\\_state](#) &\_\_obj)
- void **destroy** () throw ()
- void **init** ()

## Protected Attributes

- int **\_M\_bytes**
- int **\_M\_ext\_bom**
- [std::string](#) **\_M\_ext\_enc**
- descriptor\_type **\_M\_in\_desc**
- int **\_M\_int\_bom**
- [std::string](#) **\_M\_int\_enc**
- descriptor\_type **\_M\_out\_desc**

## 5.42.1 Detailed Description

Extension to use iconv for dealing with character encodings.

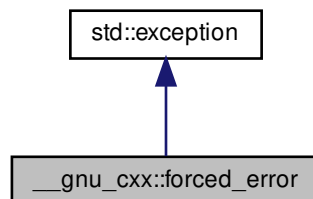
Definition at line 51 of file `codecvt_specializations.h`.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.43 \_\_gnu\_cxx::forced\_error Struct Reference

Inheritance diagram for `__gnu_cxx::forced_error`:



## Public Member Functions

- virtual const char \* **what** () const `_GLIBCXX_TXN_SAFE_DYN` noexcept



#### 5.43.1 Detailed Description

Thrown by exception safety machinery.

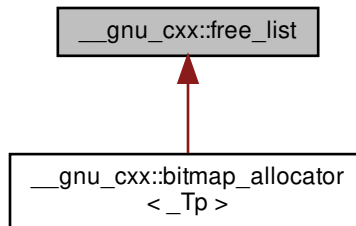
Definition at line 74 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 5.44 `__gnu_cxx::free_list` Class Reference

Inheritance diagram for `__gnu_cxx::free_list`:



##### Public Types

- typedef `__mutex` **`mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t * value_type`
- typedef `__detail::__mini_vector< value_type >` **`vector_type`**

##### Public Member Functions

- void `_M_clear` ()
- `size_t * _M_get` (size\_t `__sz`)
- void `_M_insert` (size\_t \* `__addr`) throw ()

#### 5.44.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

Definition at line 518 of file `bitmap_allocator.h`.

### 5.44.2 Member Function Documentation

#### 5.44.2.1 \_M\_clear()

```
void __gnu_cxx::free_list::_M_clear ( )
```

This function just clears the internal Free List, and gives back all the memory to the OS.

#### 5.44.2.2 \_M\_get()

```
size_t* __gnu_cxx::free_list::_M_get (
    size_t __sz )
```

This function gets a block of memory of the specified size from the free list.

##### Parameters

<code>__sz</code>	The size in bytes of the memory required.
-------------------	---

##### Returns

A pointer to the new memory block of size at least equal to that requested.

#### 5.44.2.3 \_M\_insert()

```
void __gnu_cxx::free_list::_M_insert (
    size_t * __addr ) throw ( )    [inline]
```

This function returns the block of memory to the internal free list.

##### Parameters

<code>__addr</code>	The pointer to the memory block that was given by a call to the <code>_M_get</code> function.
---------------------	---

Definition at line 628 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.45 `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

### Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Ht::const_pointer const_pointer`
- `typedef _Ht::const_reference const_reference`
- `typedef _Tp data_type`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Tp mapped_type`
- `typedef _Ht::pointer pointer`
- `typedef _Ht::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

### Public Member Functions

- `hash_map (size_type __n)`
- `hash_map (size_type __n, const hasher &__hf)`
- `hash_map (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `template<class _InputIterator >  
hash_map (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >  
hash_map (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >  
hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`
- `template<class _InputIterator >  
hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `iterator begin ()`
- `const_iterator begin () const`
- `size_type bucket_count () const`
- `void clear ()`
- `size_type count (const key_type &__key) const`
- `size_type elems_in_bucket (size_type __n) const`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `pair< iterator, iterator > equal_range (const key_type &__key)`
- `pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `size_type erase (const key_type &__key)`
- `void erase (iterator __it)`
- `void erase (iterator __f, iterator __l)`
- `iterator find (const key_type &__key)`

- `const_iterator` **find** (const `key_type` &\_\_key) const
- `allocator_type` **get\_allocator** () const
- hasher **hash\_funct** () const
- `pair`< iterator, bool > **insert** (const `value_type` &\_\_obj)
- `template<class _InputIterator>`  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- `pair`< iterator, bool > **insert\_noresize** (const `value_type` &\_\_obj)
- `key_equal` **key\_eq** () const
- `size_type` **max\_bucket\_count** () const
- `size_type` **max\_size** () const
- `_Tp` & **operator[]** (const `key_type` &\_\_key)
- void **resize** (`size_type` \_\_hint)
- `size_type` **size** () const
- void **swap** (`hash_map` &\_\_hs)

#### Friends

- `template<class _K1, class _T1, class _HF, class _EqK, class _AI>`  
bool **operator==** (const `hash_map`< \_K1, \_T1, \_HF, \_EqK, \_AI > &, const `hash_map`< \_K1, \_T1, \_HF, \_EqK, \_AI > &)

#### 5.45.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_↵
_Tp>>
class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Definition at line 83 of file `hash_map`.

The documentation for this class was generated from the following file:

- `hash_map`

## 5.46 `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

### Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Ht::const_pointer const_pointer`
- `typedef _Ht::const_reference const_reference`
- `typedef _Tp data_type`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Tp mapped_type`
- `typedef _Ht::pointer pointer`
- `typedef _Ht::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

### Public Member Functions

- `hash_multimap (size_type __n)`
- `hash_multimap (size_type __n, const hasher &__hf)`
- `hash_multimap (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `template<class _InputIterator >  
hash_multimap (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >  
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >  
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`
- `template<class _InputIterator >  
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `iterator begin ()`
- `const_iterator begin () const`
- `size_type bucket_count () const`
- `void clear ()`
- `size_type count (const key_type &__key) const`
- `size_type elems_in_bucket (size_type __n) const`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `pair< iterator, iterator > equal_range (const key_type &__key)`
- `pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `size_type erase (const key_type &__key)`
- `void erase (iterator __it)`
- `void erase (iterator __f, iterator __l)`
- `iterator find (const key_type &__key)`

- `const_iterator` **find** (`const key_type &__key`) `const`
- `allocator_type` **get\_allocator** () `const`
- hasher **hash\_funct** () `const`
- `iterator` **insert** (`const value_type &__obj`)
- `template<class _InputIterator>`  
`void insert` (`_InputIterator __f, _InputIterator __l`)
- `iterator` **insert\_noresize** (`const value_type &__obj`)
- `key_equal` **key\_eq** () `const`
- `size_type` **max\_bucket\_count** () `const`
- `size_type` **max\_size** () `const`
- `void` **resize** (`size_type __hint`)
- `size_type` **size** () `const`
- `void` **swap** (`hash_multimap &__hs`)

#### Friends

- `template<class _K1, class _T1, class _HF, class _EqK, class _AI>`  
`bool operator==` (`const hash_multimap<_K1, _T1, _HF, _EqK, _AI> &, const hash_multimap<_K1, _T1, _HF, _EqK, _AI> &`)

#### 5.46.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>
```

This is an SGI extension.

**Todo** \nNeeds documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-style.html>

Definition at line 296 of file `hash_map`.

The documentation for this class was generated from the following file:

- `hash_map`

## 5.47 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>` Class Template Reference

#### Public Types

- `typedef _Ht::allocator_type` **allocator\_type**
- `typedef _Ht::const_iterator` **const\_iterator**
- `typedef _Alloc::const_pointer` **const\_pointer**
- `typedef _Alloc::const_reference` **const\_reference**
- `typedef _Ht::difference_type` **difference\_type**
- `typedef _Ht::hasher` **hasher**
- `typedef _Ht::const_iterator` **iterator**
- `typedef _Ht::key_equal` **key\_equal**
- `typedef _Ht::key_type` **key\_type**
- `typedef _Alloc::pointer` **pointer**
- `typedef _Alloc::reference` **reference**
- `typedef _Ht::size_type` **size\_type**
- `typedef _Ht::value_type` **value\_type**

## Public Member Functions

- **hash\_multiset** (size\_type \_\_n)
- **hash\_multiset** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_it)
- void **erase** (iterator \_\_f, iterator \_\_l)
- iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- iterator **insert** (const value\_type &\_\_obj)
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- iterator **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** ([hash\\_multiset](#) &hs)

## Friends

- template<class \_Val, class \_HF, class \_EqK, class \_AI >  
bool **operator==** (const [hash\\_multiset](#)< \_Val, \_HF, \_EqK, \_AI > &, const [hash\\_multiset](#)< \_Val, \_HF, \_EqK, \_AI > &)

## 5.47.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html)

Definition at line 285 of file `hash_set`.

The documentation for this class was generated from the following file:

- [hash\\_set](#)

5.48 `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >` Class Template Reference

## Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Alloc::const_pointer` **const\_pointer**
- typedef `_Alloc::const_reference` **const\_reference**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::const_iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Alloc::pointer` **pointer**
- typedef `_Alloc::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

## Public Member Functions

- **hash\_set** (size\_type \_\_n)
- **hash\_set** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_set** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eql, const allocator\_type &\_\_a=allocator\_type())
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)



- `template<class _InputIterator >`  
`hash_set` (`_InputIterator __f`, `_InputIterator __l`, `size_type __n`, `const hasher &__hf`, `const key_equal &__eq`,  
`const allocator_type &__a=allocator_type()`)
- `iterator begin` () `const`
- `size_type bucket_count` () `const`
- `void clear` ()
- `size_type count` (`const key_type &__key`) `const`
- `size_type elems_in_bucket` (`size_type __n`) `const`
- `bool empty` () `const`
- `iterator end` () `const`
- `pair< iterator, iterator > equal_range` (`const key_type &__key`) `const`
- `size_type erase` (`const key_type &__key`)
- `void erase` (`iterator __it`)
- `void erase` (`iterator __f`, `iterator __l`)
- `iterator find` (`const key_type &__key`) `const`
- `allocator_type get_allocator` () `const`
- `hasher hash_func` () `const`
- `pair< iterator, bool > insert` (`const value_type &__obj`)
- `template<class _InputIterator >`  
`void insert` (`_InputIterator __f`, `_InputIterator __l`)
- `pair< iterator, bool > insert_noresize` (`const value_type &__obj`)
- `key_equal key_eq` () `const`
- `size_type max_bucket_count` () `const`
- `size_type max_size` () `const`
- `void resize` (`size_type __hint`)
- `size_type size` () `const`
- `void swap` (`hash_set &__hs`)

#### Friends

- `template<class _Val, class _HF, class _EqK, class _AI >`  
`bool operator==` (`const hash_set< _Val, _HF, _EqK, _AI > &`, `const hash_set< _Val, _HF, _EqK, _AI > &`)

#### 5.48.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

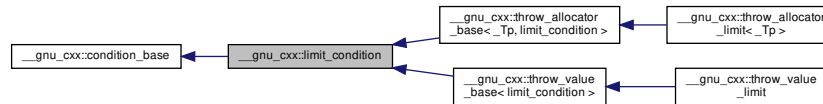
Definition at line 84 of file `hash_set`.

The documentation for this class was generated from the following file:

- `hash_set`

5.49 `__gnu_cxx::limit_condition` Struct Reference

Inheritance diagram for `__gnu_cxx::limit_condition`:



## Classes

- struct [always\\_adjustor](#)
- struct [limit\\_adjustor](#)
- struct [never\\_adjustor](#)

## Static Public Member Functions

- static `size_t` & **count** ()
- static `size_t` & **limit** ()
- static void **set\_limit** (const `size_t` \_\_l)
- static void **throw\_conditionally** ()

## 5.49.1 Detailed Description

Base class for incremental control and throw.

Definition at line 412 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.50 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

## 5.50.1 Detailed Description

Always enter the condition.

Definition at line 436 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.51 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### Public Member Functions

- **`limit_adjustor`** (`const size_t __l`)

#### 5.51.1 Detailed Description

Enter the nth condition.

Definition at line 442 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.52 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

#### 5.52.1 Detailed Description

Never enter the condition.

Definition at line 430 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.53 `__gnu_cxx::malloc_allocator<_Tp>` Class Template Reference

### Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true\_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- **malloc\_allocator** (const [malloc\\_allocator](#) &) noexcept
- template<typename `_Tp1` >  
**malloc\_allocator** (const [malloc\\_allocator](#)< `_Tp1` > &) noexcept
- pointer **address** (reference `__x`) const noexcept
- const\_pointer **address** (const\_reference `__x`) const noexcept
- pointer **allocate** (size\_type `__n`, const void \*`=0`)
- template<typename `_Up` , typename... `_Args`>  
void **construct** (`_Up` \*`__p`, `_Args` &&... `__args`)
- void **deallocate** (pointer `__p`, size\_type)
- template<typename `_Up` >  
void **destroy** (`_Up` \*`__p`)
- size\_type **max\_size** () const noexcept

## 5.53.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::malloc_allocator<_Tp>
```

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

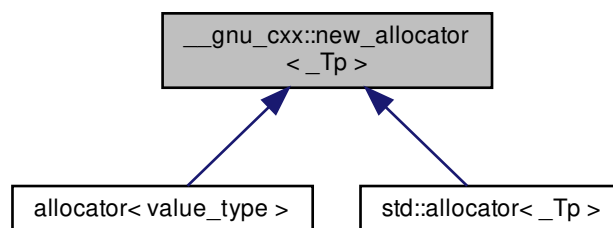
Definition at line 57 of file `malloc_allocator.h`.

The documentation for this class was generated from the following file:

- [malloc\\_allocator.h](#)

5.54 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **new\_allocator** (const [new\\_allocator](#) &) noexcept
- template<typename \_Tp1 >  
  **new\_allocator** (const [new\\_allocator](#)< \_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \* \_\_p = static\_cast< const void \* >(0))
- template<typename \_Up, typename... \_Args>  
  void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type)
- template<typename \_Up >  
  void **destroy** (\_Up \* \_\_p)
- size\_type **max\_size** () const noexcept

#### 5.54.1 Detailed Description

```
template<typename _Tp>  
class __gnu_cxx::new_allocator< _Tp >
```

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

### Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 58 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new\\_allocator.h](#)

## 5.55 `__gnu_cxx::project1st<_Arg1, _Arg2 >` Struct Template Reference

Inherits `__gnu_cxx::_Project1st<_Arg1, _Arg2 >`.

### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `_Arg1 operator()` (const `_Arg1` &`__x`, const `_Arg2` &) const

#### 5.55.1 Detailed Description

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project1st<_Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 237 of file `ext/functional`.

#### 5.55.2 Member Typedef Documentation

##### 5.55.2.1 `first_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary\_function<\_Arg1, \_Arg2, \_Result >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 5.55.2.2 `result_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary\_function<\_Arg1, \_Arg2, \_Result >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

### 5.55.2.3 second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary\_function< _Arg1, _Arg2, _Result >::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.56 \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 > Struct Template Reference

Inherits `__gnu_cxx::Project2nd< _Arg1, _Arg2 >`.

### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `_Arg2 operator()` (`const _Arg1 &`, `const _Arg2 &__y`) `const`

### 5.56.1 Detailed Description

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 241 of file `ext/functional`.

### 5.56.2 Member Typedef Documentation

## 5.56.2.1 first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

## 5.56.2.2 result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 5.56.2.3 second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

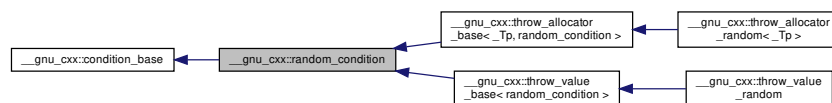
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.57 \_\_gnu\_cxx::random\_condition Struct Reference

Inheritance diagram for \_\_gnu\_cxx::random\_condition:





## Classes

- struct [always\\_adjustor](#)
- struct [group\\_adjustor](#)
- struct [never\\_adjustor](#)

## Public Member Functions

- void **seed** (unsigned long \_\_s)

## Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### 5.57.1 Detailed Description

Base class for random probability control and throw.

Definition at line 484 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.58 \_\_gnu\_cxx::random\_condition::always\_adjustor Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### 5.58.1 Detailed Description

Always enter the condition.

Definition at line 517 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.59 \_\_gnu\_cxx::random\_condition::group\_adjustor Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

## Public Member Functions

- `group_adjustor` (size\_t size)

## 5.59.1 Detailed Description

Group condition.

Definition at line 502 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.60 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

## 5.60.1 Detailed Description

Never enter the condition.

Definition at line 511 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.61 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >`.

## Public Types

- `typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rb_tree_iterator< value_type > iterator`
- `typedef _Key key_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _Val value_type`

## Public Member Functions

- **rb\_tree** (const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- bool **\_\_rb\_verify** () const
- template<typename \_Iterator >  
void **\_M\_assign\_equal** (\_Iterator, \_Iterator)
- template<typename \_Iterator >  
void **\_M\_assign\_unique** (\_Iterator, \_Iterator)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
size\_type **\_M\_count\_tr** (const \_Kt &\_\_k) const
- template<typename... \_Args>  
iterator **\_M\_emplace\_equal** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **\_M\_emplace\_hint\_equal** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **\_M\_emplace\_hint\_unique** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
pair< iterator, bool > **\_M\_emplace\_unique** (\_Args &&... \_\_args)
- template<typename... \_Args>  
pair< typename \_Rb\_tree< \_Key, \_Val, \_KeyOfValue, \_Compare, \_Alloc >::iterator, bool > **\_M\_emplace\_←  
unique** (\_Args &&... \_\_args)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
pair< iterator, iterator > **\_M\_equal\_range\_tr** (const \_Kt &\_\_k)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
pair< const\_iterator, const\_iterator > **\_M\_equal\_range\_tr** (const \_Kt &\_\_k) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
iterator **\_M\_find\_tr** (const \_Kt &\_\_k)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
const\_iterator **\_M\_find\_tr** (const \_Kt &\_\_k) const
- pair< \_Base\_ptr, \_Base\_ptr > **\_M\_get\_insert\_equal\_pos** (const key\_type &\_\_k)
- pair< \_Base\_ptr, \_Base\_ptr > **\_M\_get\_insert\_hint\_equal\_pos** (const\_iterator \_\_pos, const key\_type &\_\_k)
- pair< \_Base\_ptr, \_Base\_ptr > **\_M\_get\_insert\_hint\_unique\_pos** (const\_iterator \_\_pos, const key\_type &\_\_k)
- pair< \_Base\_ptr, \_Base\_ptr > **\_M\_get\_insert\_unique\_pos** (const key\_type &\_\_k)
- \_Node\_allocator & **\_M\_get\_Node\_allocator** () noexcept
- const \_Node\_allocator & **\_M\_get\_Node\_allocator** () const noexcept
- template<typename \_Arg >  
iterator **\_M\_insert\_equal** (\_Arg &&\_\_x)
- template<typename \_InputIterator >  
void **\_M\_insert\_equal** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<class \_II >  
void **\_M\_insert\_equal** (\_II \_\_first, \_II \_\_last)
- template<typename \_Arg, typename \_NodeGen >  
iterator **\_M\_insert\_equal\_** (const\_iterator \_\_pos, \_Arg &&\_\_x, \_NodeGen &)
- template<typename \_Arg >  
iterator **\_M\_insert\_equal\_** (const\_iterator \_\_pos, \_Arg &&\_\_x)
- template<typename \_Arg >  
pair< iterator, bool > **\_M\_insert\_unique** (\_Arg &&\_\_x)
- template<typename \_InputIterator >  
void **\_M\_insert\_unique** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Arg >  
pair< typename \_Rb\_tree< \_Key, \_Val, \_KeyOfValue, \_Compare, \_Alloc >::iterator, bool > **\_M\_insert\_unique**  
( \_Arg &&\_\_v)

- `template<class _II >`  
`void _M_insert_unique (_II __first, _II __last)`
- `template<typename _Arg, typename _NodeGen >`  
`iterator _M_insert_unique (const_iterator __pos, _Arg &&__x, _NodeGen &)`
- `template<typename _Arg >`  
`iterator _M_insert_unique (const_iterator __pos, _Arg &&__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator _M_lower_bound_tr (const _Kt &__k)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator _M_lower_bound_tr (const _Kt &__k) const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator _M_upper_bound_tr (const _Kt &__k)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator _M_upper_bound_tr (const _Kt &__k) const`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `void clear () noexcept`
- `size_type count (const key_type &__k) const`
- `bool empty () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `pair< iterator, iterator > equal_range (const key_type &__k)`
- `pair< const_iterator, const_iterator > equal_range (const key_type &__k) const`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __position)`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (iterator __position)`
- `size_type erase (const key_type &__x)`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __first, const_iterator __last)`
- `void erase (const key_type *__first, const key_type *__last)`
- `iterator find (const key_type &__k)`
- `const_iterator find (const key_type &__k) const`
- `allocator_type get_allocator () const noexcept`
- `_Compare key_comp () const`
- `iterator lower_bound (const key_type &__k)`
- `const_iterator lower_bound (const key_type &__k) const`
- `size_type max_size () const noexcept`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `size_type size () const noexcept`
- `void swap (_Rb_tree &__t) noexcept(/*conditional */)`
- `iterator upper_bound (const key_type &__k)`
- `const_iterator upper_bound (const key_type &__k) const`

## Protected Types

- `typedef _Rb_tree_node_base * _Base_ptr`
- `typedef const _Rb_tree_node_base * _Const_Base_ptr`
- `typedef const _Rb_tree_node< _Val > * _Const_Link_type`
- `typedef _Rb_tree_node< _Val > * _Link_type`

### Protected Member Functions

- `_Link_type _M_begin () noexcept`
- `_Const_Link_type _M_begin () const noexcept`
- `template<typename _NodeGen >  
_Link_type _M_clone_node (_Const_Link_type __x, _NodeGen &__node_gen)`
- `template<typename... _Args>  
void _M_construct_node (_Link_type __node, _Args &&... __args)`
- `template<typename... _Args>  
_Link_type _M_create_node (_Args &&... __args)`
- `void _M_destroy_node (_Link_type __p) noexcept`
- `void _M_drop_node (_Link_type __p) noexcept`
- `_Base_ptr _M_end () noexcept`
- `_Const_Base_ptr _M_end () const noexcept`
- `_Link_type _M_get_node ()`
- `_Base_ptr & _M_leftmost () noexcept`
- `_Const_Base_ptr _M_leftmost () const noexcept`
- `void _M_put_node (_Link_type __p) noexcept`
- `_Base_ptr & _M_rightmost () noexcept`
- `_Const_Base_ptr _M_rightmost () const noexcept`
- `_Base_ptr & _M_root () noexcept`
- `_Const_Base_ptr _M_root () const noexcept`

### Static Protected Member Functions

- `static const _Key & _S_key (_Const_Link_type __x)`
- `static const _Key & _S_key (_Const_Base_ptr __x)`
- `static _Link_type _S_left (_Base_ptr __x) noexcept`
- `static _Const_Link_type _S_left (_Const_Base_ptr __x) noexcept`
- `static _Base_ptr _S_maximum (_Base_ptr __x) noexcept`
- `static _Const_Base_ptr _S_maximum (_Const_Base_ptr __x) noexcept`
- `static _Base_ptr _S_minimum (_Base_ptr __x) noexcept`
- `static _Const_Base_ptr _S_minimum (_Const_Base_ptr __x) noexcept`
- `static _Link_type _S_right (_Base_ptr __x) noexcept`
- `static _Const_Link_type _S_right (_Const_Base_ptr __x) noexcept`
- `static const_reference _S_value (_Const_Link_type __x)`
- `static const_reference _S_value (_Const_Base_ptr __x)`

### Protected Attributes

- `_Rb_tree_impl<_Compare> _M_impl`

## 5.61.1 Detailed Description

```
template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = allocator<_Value>>
struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<_style.html)

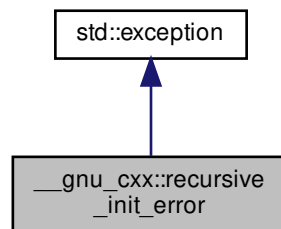
Definition at line 80 of file rb\_tree.

The documentation for this struct was generated from the following file:

- [rb\\_tree](#)

## 5.62 \_\_gnu\_cxx::recursive\_init\_error Class Reference

Inheritance diagram for \_\_gnu\_cxx::recursive\_init\_error:



## Public Member Functions

- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.62.1 Detailed Description

Exception thrown by \_\_cxa\_guard\_acquire.

C++ 2011 6.7 [stmt.dcl]/4: If control re-enters the declaration recursively while the variable is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 695 of file cxxabi.h.

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

### 5.63 `__gnu_cxx::rope<_CharT, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::Rope_base<_CharT, _Alloc>`.

#### Public Types

- `typedef _Rope_RopeConcatenation<_CharT, _Alloc> __C`
- `typedef _Rope_RopeFunction<_CharT, _Alloc> __F`
- `typedef _Rope_RopeLeaf<_CharT, _Alloc> __L`
- `typedef _Rope_RopeSubstring<_CharT, _Alloc> __S`
- `typedef _Alloc::template rebind<__C>::other _CAlloc`
- `typedef _Alloc::template rebind<_CharT>::other _DataAlloc`
- `typedef _Alloc::template rebind<__F>::other _FAlloc`
- `typedef _Alloc::template rebind<__L>::other _LAlloc`
- `typedef _Alloc::template rebind<__S>::other _SAlloc`
- `typedef _Rope_const_iterator<_CharT, _Alloc> const_iterator`
- `typedef const _CharT * const_pointer`
- `typedef _CharT const_reference`
- `typedef std::reverse\_iterator<const_iterator> const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rope_iterator<_CharT, _Alloc> iterator`
- `typedef _Rope_char_ptr_proxy<_CharT, _Alloc> pointer`
- `typedef _Rope_char_ref_proxy<_CharT, _Alloc> reference`
- `typedef std::reverse\_iterator<iterator> reverse_iterator`
- `typedef size_t size_type`
- `typedef _CharT value_type`

#### Public Member Functions

- **rope** (const \_CharT \* \_\_s, const allocator\_type & \_\_a=allocator\_type())
- **rope** (const \_CharT \* \_\_s, size\_t \_\_len, const allocator\_type & \_\_a=allocator\_type())
- **rope** (const \_CharT \* \_\_s, const \_CharT \* \_\_e, const allocator\_type & \_\_a=allocator\_type())
- **rope** (const const\_iterator & \_\_s, const const\_iterator & \_\_e, const allocator\_type & \_\_a=allocator\_type())
- **rope** (const iterator & \_\_s, const iterator & \_\_e, const allocator\_type & \_\_a=allocator\_type())
- **rope** (\_CharT \_\_c, const allocator\_type & \_\_a=allocator\_type())
- **rope** (size\_t \_\_n, \_CharT \_\_c, const allocator\_type & \_\_a=allocator\_type())
- **rope** (const allocator\_type & \_\_a=allocator\_type())
- **rope** (char\_producer<\_CharT> \* \_\_fn, size\_t \_\_len, bool \_\_delete\_fn, const allocator\_type & \_\_a=allocator\_type())
- **rope** (const [rope](#) & \_\_x, const allocator\_type & \_\_a=allocator\_type())
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- [rope](#) & **append** (const \_CharT \* \_\_iter, size\_t \_\_n)
- [rope](#) & **append** (const \_CharT \* \_\_c\_string)
- [rope](#) & **append** (const \_CharT \* \_\_s, const \_CharT \* \_\_e)
- [rope](#) & **append** (const\_iterator \_\_s, const\_iterator \_\_e)
- [rope](#) & **append** (\_CharT \_\_c)
- [rope](#) & **append** ()
- [rope](#) & **append** (const [rope](#) & \_\_y)

- `rope` & **append** (size\_t \_\_n, \_CharT \_\_c)
- void **apply\_to\_pieces** (size\_t \_\_begin, size\_t \_\_end, \_Rope\_char\_consumer<\_CharT> &\_\_c) const
- \_CharT **at** (size\_type \_\_pos) const
- \_CharT **back** () const
- void **balance** ()
- const\_iterator **begin** () const
- const\_iterator **begin** ()
- const \_CharT \* **c\_str** () const
- void **clear** ()
- int **compare** (const `rope` &\_\_y) const
- const\_iterator **const\_begin** () const
- const\_iterator **const\_end** () const
- `const_reverse_iterator` **const\_rbegin** () const
- `const_reverse_iterator` **const\_rend** () const
- void **copy** (\_CharT \*\_\_buffer) const
- size\_type **copy** (size\_type \_\_pos, size\_type \_\_n, \_CharT \*\_\_buffer) const
- void **delete\_c\_str** ()
- void **dump** ()
- bool **empty** () const
- const\_iterator **end** () const
- const\_iterator **end** ()
- void **erase** (size\_t \_\_p, size\_t \_\_n)
- void **erase** (size\_t \_\_p)
- iterator **erase** (const iterator &\_\_p, const iterator &\_\_q)
- iterator **erase** (const iterator &\_\_p)
- size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- \_CharT **front** () const
- allocator\_type **get\_allocator** () const
- void **insert** (size\_t \_\_p, const `rope` &\_\_r)
- void **insert** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **insert** (size\_t \_\_p, \_CharT \_\_c)
- void **insert** (size\_t \_\_p)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **insert** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const `rope` &\_\_r)
- iterator **insert** (const iterator &\_\_p, size\_t \_\_n, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*c\_string)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const iterator &\_\_i, const iterator &\_\_j)
- size\_type **length** () const
- size\_type **max\_size** () const
- iterator **mutable\_begin** ()
- iterator **mutable\_end** ()



- [reverse\\_iterator](#) **mutable\_rbegin** ()
- reference **mutable\_reference\_at** (size\_type \_\_pos)
- [reverse\\_iterator](#) **mutable\_rend** ()
- [rope](#) & **operator=** (const [rope](#) &\_\_x)
- \_CharT **operator[]** (size\_type \_\_pos) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_CharT \_\_x)
- void **push\_front** (\_CharT \_\_x)
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [const\\_reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- [const\\_reverse\\_iterator](#) **rend** ()
- void **replace** (size\_t \_\_p, size\_t \_\_n, const [rope](#) &\_\_r)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_t \_\_p, \_CharT \_\_c)
- void **replace** (size\_t \_\_p, const [rope](#) &\_\_r)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const [rope](#) &\_\_r)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_t \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const [rope](#) &\_\_r)
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const\_iterator \_\_i, const\_iterator \_\_j)
- void **replace** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- const \_CharT \* **replace\_with\_c\_str** ()
- size\_type **size** () const
- [rope](#) **substr** (size\_t \_\_start, size\_t \_\_len=1) const
- [rope](#) **substr** (iterator \_\_start, iterator \_\_end) const
- [rope](#) **substr** (iterator \_\_start) const
- [rope](#) **substr** (const\_iterator \_\_start, const\_iterator \_\_end) const
- [rope](#)< \_CharT, \_Alloc > **substr** (const\_iterator \_\_start)
- void **swap** ([rope](#) &\_\_b)

## Static Public Member Functions

- static `__C * _C_allocate` (`size_t __n`)
- static void `_C_deallocate` (`__C * __p, size_t __n`)
- static `_CharT * _Data_allocate` (`size_t __n`)
- static void `_Data_deallocate` (`_CharT * __p, size_t __n`)
- static `__F * _F_allocate` (`size_t __n`)
- static void `_F_deallocate` (`__F * __p, size_t __n`)
- static `__L * _L_allocate` (`size_t __n`)
- static void `_L_deallocate` (`__L * __p, size_t __n`)
- static `__S * _S_allocate` (`size_t __n`)
- static void `_S_deallocate` (`__S * __p, size_t __n`)

## Public Attributes

- `_RopeRep * _M_tree_ptr`

## Static Public Attributes

- static const `size_type npos`

## Protected Types

- enum { `_S_copy_max` }
- typedef `_Rope_base<_CharT, _Alloc>` `_Base`
- typedef `_CharT * _Cstrptr`
- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` `_RopeConcatenation`
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` `_RopeFunction`
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` `_RopeLeaf`
- typedef `_Rope_RopeRep<_CharT, _Alloc>` `_RopeRep`
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` `_RopeSubstring`
- typedef `_Rope_self_destruct_ptr<_CharT, _Alloc>` `_Self_destruct_ptr`
- typedef `_Base::allocator_type` `allocator_type`

## Static Protected Member Functions

- static `size_t _S_allocated_capacity` (`size_t __n`)
- static bool `_S_apply_to_pieces` (`_Rope_char_consumer<_CharT> &__c, const _RopeRep * __r, size_t __begin, size_t __end`)
- static `_RopeRep * _S_concat` (`_RopeRep * __left, _RopeRep * __right`)
- static `_RopeRep * _S_concat_char_iter` (`_RopeRep * __r, const _CharT * __iter, size_t __slen`)
- static `_RopeRep * _S_destr_concat_char_iter` (`_RopeRep * __r, const _CharT * __iter, size_t __slen`)
- static `_RopeLeaf * _S_destr_leaf_concat_char_iter` (`_RopeLeaf * __r, const _CharT * __iter, size_t __slen`)
- static `_CharT _S_fetch` (`_RopeRep * __r, size_type __pos`)
- static `_CharT * _S_fetch_ptr` (`_RopeRep * __r, size_type __pos`)
- static bool `_S_is0` (`_CharT __c`)
- static `_RopeLeaf * _S_leaf_concat_char_iter` (`_RopeLeaf * __r, const _CharT * __iter, size_t __slen`)

- static `_RopeConcatenation * _S_new_RopeConcatenation` (`_RopeRep *__left`, `_RopeRep *__right`, `allocator_type &__a`)
- static `_RopeFunction * _S_new_RopeFunction` (`char_producer< _CharT > *__f`, `size_t __size`, `bool __d`, `allocator_type &__a`)
- static `_RopeLeaf * _S_new_RopeLeaf` (`_CharT *__s`, `size_t __size`, `allocator_type &__a`)
- static `_RopeSubstring * _S_new_RopeSubstring` (`_Rope_RopeRep< _CharT, _Alloc > *__b`, `size_t __l`, `size_t __l`, `allocator_type &__a`)
- static void `_S_ref` (`_RopeRep *__t`)
- static `_RopeLeaf * _S_RopeLeaf_from_unowned_char_ptr` (`const _CharT *__s`, `size_t __size`, `allocator_type &__a`)
- static `size_t _S_rounded_up_size` (`size_t __n`)
- static `_RopeRep * _S_substring` (`_RopeRep *__base`, `size_t __start`, `size_t __endp1`)
- static `_RopeRep * _S_tree_concat` (`_RopeRep *__left`, `_RopeRep *__right`)
- static void `_S_unref` (`_RopeRep *__t`)
- static `_RopeRep * replace` (`_RopeRep *__old`, `size_t __pos1`, `size_t __pos2`, `_RopeRep *__r`)

#### Static Protected Attributes

- static `_CharT _S_empty_c_str` [1]

#### Friends

- class `_Rope_char_ptr_proxy< _CharT, _Alloc >`
- class `_Rope_char_ref_proxy< _CharT, _Alloc >`
- class `_Rope_const_iterator< _CharT, _Alloc >`
- class `_Rope_iterator< _CharT, _Alloc >`
- class `_Rope_iterator_base< _CharT, _Alloc >`
- struct `_Rope_RopeRep< _CharT, _Alloc >`
- struct `_Rope_RopeSubstring< _CharT, _Alloc >`
- template<class `_CharT2`, class `_Alloc2` >  
`rope< _CharT2, _Alloc2 > operator+ (const rope< _CharT2, _Alloc2 > &__left, const rope< _CharT2, _Alloc2 > &__right)`
- template<class `_CharT2`, class `_Alloc2` >  
`rope< _CharT2, _Alloc2 > operator+ (const rope< _CharT2, _Alloc2 > &__left, const _CharT2 *__right)`
- template<class `_CharT2`, class `_Alloc2` >  
`rope< _CharT2, _Alloc2 > operator+ (const rope< _CharT2, _Alloc2 > &__left, _CharT2 __right)`

#### 5.63.1 Detailed Description

```
template<class _CharT, class _Alloc>
class __gnu_cxx::rope< _CharT, _Alloc >
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html)

Definition at line 327 of file rope.

The documentation for this class was generated from the following files:

- [rope](#)
- [ropeimpl.h](#)

## 5.64 `__gnu_cxx::select1st<_Pair>` Struct Template Reference

Inherits `std::_Select1st<_Pair>`.

### Public Types

- typedef `_Pair` [argument\\_type](#)
- typedef `_Pair::first_type` [result\\_type](#)

### Public Member Functions

- `_Pair::first_type & operator() (_Pair &__x) const`
- `const _Pair::first_type & operator() (const _Pair &__x) const`
- `template<typename _Pair2 > _Pair2::first_type & operator() (_Pair2 &__x) const`
- `template<typename _Pair2 > const _Pair2::first_type & operator() (const _Pair2 &__x) const`

#### 5.64.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select1st<_Pair>
```

An [SGI extension](#) .

Definition at line 200 of file `ext/functional`.

#### 5.64.2 Member Typedef Documentation

##### 5.64.2.1 `argument_type`

```
typedef _Pair std::unary\_function<_Pair, _Pair::first_type>::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

### 5.64.2.2 result\_type

```
typedef _Pair::first_type std::unary_function< _Pair , _Pair::first_type >::result_type [inherited]
```

result\_type is the return type

Definition at line 111 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.65 \_\_gnu\_cxx::select2nd< \_Pair > Struct Template Reference

Inherits std::\_Select2nd< \_Pair >.

### Public Types

- typedef \_Pair [argument\\_type](#)
- typedef \_Pair::second\_type [result\\_type](#)

### Public Member Functions

- \_Pair::second\_type & **operator()** (\_Pair &\_\_x) const
- const \_Pair::second\_type & **operator()** (const \_Pair &\_\_x) const

### 5.65.1 Detailed Description

```
template<class _Pair>  
struct __gnu_cxx::select2nd< _Pair >
```

An [SGI extension](#) .

Definition at line 205 of file ext/functional.

### 5.65.2 Member Typedef Documentation

## 5.65.2.1 argument\_type

```
typedef _Pair std::unary_function< _Pair , _Pair::second_type >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 5.65.2.2 result\_type

```
typedef _Pair::second_type std::unary_function< _Pair , _Pair::second_type >::result_type [inherited]
```

result\_type is the return type

Definition at line 111 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.66 \_\_gnu\_cxx::slist&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inherits \_\_gnu\_cxx::\_Slist\_base< \_Tp, \_Alloc >.

## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_Slist\_iterator< \_Tp, const \_Tp &, const \_Tp \* > **const\_iterator**
- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Slist\_iterator< \_Tp, \_Tp &, \_Tp \* > **iterator**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **slist** (const allocator\_type &\_\_a=allocator\_type())
- **slist** (size\_type \_\_n, const value\_type &\_\_x, const allocator\_type &\_\_a=allocator\_type())
- **slist** (size\_type \_\_n)
- template<class \_InputIterator >  
**slist** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- **slist** (const [slist](#) &\_\_x)
- template<class \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<class \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_fill\_assign** (size\_type \_\_n, const Tp &\_\_val)
- void **assign** (size\_type \_\_n, const Tp &\_\_val)
- template<class \_InputIterator >  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **before\_begin** ()
- const\_iterator **before\_begin** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_pos)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase\_after** (iterator \_\_pos)
- iterator **erase\_after** (iterator \_\_before\_first, iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_pos, const value\_type &\_\_x)
- iterator **insert** (iterator \_\_pos)
- void **insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos, const value\_type &\_\_x)
- iterator **insert\_after** (iterator \_\_pos)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- size\_type **max\_size** () const
- void **merge** ([slist](#) &\_\_x)
- template<class \_StrictWeakOrdering >  
void **merge** ([slist](#) &, \_StrictWeakOrdering)
- [slist](#) & **operator=** (const [slist](#) &\_\_x)
- void **pop\_front** ()
- iterator **previous** (const\_iterator \_\_pos)
- const\_iterator **previous** (const\_iterator \_\_pos) const
- void **push\_front** (const value\_type &\_\_x)
- void **push\_front** ()

- void **remove** (const \_Tp &\_\_val)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **resize** (size\_type new\_size)
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<class \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **splice** (iterator \_\_pos, [slist](#) &\_\_x)
- void **splice** (iterator \_\_pos, [slist](#) &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_pos, [slist](#) &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice\_after** (iterator \_\_pos, iterator \_\_before\_first, iterator \_\_before\_last)
- void **splice\_after** (iterator \_\_pos, iterator \_\_prev)
- void **splice\_after** (iterator \_\_pos, [slist](#) &\_\_x)
- void **swap** ([slist](#) &\_\_x)
- void **unique** ()
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_pred)

#### Private Types

- typedef \_Alloc::template rebind<\_Slist\_node<\_Tp>>::other **\_Node\_alloc**

#### Private Member Functions

- \_Slist\_node\_base \* **\_M\_erase\_after** (\_Slist\_node\_base \*\_\_pos)
- \_Slist\_node\_base \* **\_M\_erase\_after** (\_Slist\_node\_base \*, \_Slist\_node\_base \*)
- \_Slist\_node<\_Tp> \* **\_M\_get\_node** ()
- void **\_M\_put\_node** (\_Slist\_node<\_Tp> \*\_\_p)

#### Private Attributes

- \_Slist\_node\_base **\_M\_head**

#### 5.66.1 Detailed Description

```
template<class _Tp, class _Alloc = allocator<_Tp>>
class __gnu_cxx::slist<_Tp, _Alloc>
```

This is an SGI extension.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 292 of file slist.

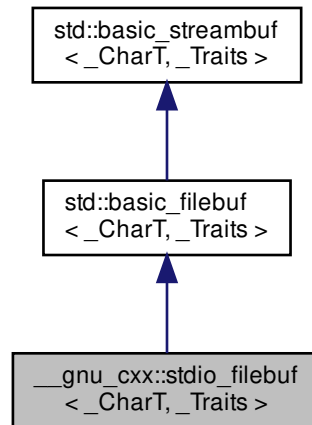
The documentation for this class was generated from the following file:

- [slist](#)



## 5.67 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



### Public Types

- `typedef codecvt< char_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic_filebuf< char_type, traits_type > __filebuf_type`
- `typedef traits_type::state_type __state_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef std::size_t size_t`
- `typedef _Traits traits_type`

### Public Member Functions

- `stdio_filebuf ()`
- `stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
- `stdio_filebuf (std::__c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
- `stdio_filebuf (stdio_filebuf &&)=default`
- `virtual ~stdio_filebuf ()`
- `__filebuf_type * close ()`
- `int fd ()`
- `std::__c_file * file ()`

- locale `getloc` () const
- streamsize `in_avail` ()
- bool `is_open` () const throw ()
- `__filebuf_type` \* `open` (const char \* \_\_s, ios\_base::openmode \_\_mode)
- `__filebuf_type` \* `open` (const std::string & \_\_s, ios\_base::openmode \_\_mode)
- `stdio_filebuf` & `operator=` (`stdio_filebuf` &&)=default
- locale `pubimbue` (const locale & \_\_loc)
- int\_type `sbumpc` ()
- int\_type `sgetc` ()
- streamsize `sgetn` (char\_type \* \_\_s, streamsize \_\_n)
- int\_type `snextc` ()
- int\_type `sputbackc` (char\_type \_\_c)
- int\_type `sputc` (char\_type \_\_c)
- streamsize `sputn` (const char\_type \* \_\_s, streamsize \_\_n)
- int\_type `sungetc` ()
- void `swap` (`stdio_filebuf` & \_\_fb)
- void `swap` (`basic_filebuf` &)
- `basic_streambuf` \* `pubsetbuf` (char\_type \* \_\_s, streamsize \_\_n)
- pos\_type `pubseekoff` (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type `pubseekpos` (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int `pubsync` ()

#### Protected Member Functions

- void `__safe_gbump` (streamsize \_\_n)
- void `__safe_pbump` (streamsize \_\_n)
- void `_M_allocate_internal_buffer` ()
- bool `_M_convert_to_external` (char\_type \*, streamsize)
- void `_M_create_pback` ()
- void `_M_destroy_internal_buffer` () throw ()
- void `_M_destroy_pback` () throw ()
- int `_M_get_ext_pos` (\_\_state\_type & \_\_state)
- pos\_type `_M_seek` (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
- void `_M_set_buffer` (streamsize \_\_off)
- bool `_M_terminate_output` ()
- void `gbump` (int \_\_n)
- virtual void `imbue` (const locale & \_\_loc)
- virtual void `imbue` (const locale & \_\_loc \_\_attribute\_\_((\_\_unused\_\_)))
- virtual int\_type `overflow` (int\_type \_\_c=\_\_Traits::eof())
- virtual int\_type `overflow` (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
- virtual int\_type `pbackfail` (int\_type \_\_c=\_\_Traits::eof())
- virtual int\_type `pbackfail` (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
- void `pbump` (int \_\_n)
- virtual pos\_type `seekoff` (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)

- virtual pos\_type [seekpos](#) (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) (char\_type \* \_\_s, streamsize \_\_n)
- void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual streamsize [showmanyc](#) ()
- void **swap** ([basic\\_streambuf](#) & \_\_sb)
- virtual int [sync](#) ()
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- virtual streamsize [xsgetn](#) (char\_type \* \_\_s, streamsize \_\_n)
- virtual streamsize [xspn](#) (const char\_type \* \_\_s, streamsize \_\_n)

- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const

- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

#### Protected Attributes

- char\_type \* [\\_M\\_buf](#)
- bool [\\_M\\_buf\\_allocated](#)
- locale [\\_M\\_buf\\_locale](#)
- size\_t [\\_M\\_buf\\_size](#)
- const [\\_\\_codecvt\\_type](#) \* [\\_M\\_codecvt](#)
- char \* [\\_M\\_ext\\_buf](#)
- streamsize [\\_M\\_ext\\_buf\\_size](#)
- char \* [\\_M\\_ext\\_end](#)
- const char \* [\\_M\\_ext\\_next](#)
- [\\_\\_file\\_type](#) [\\_M\\_file](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- [\\_\\_c\\_lock](#) [\\_M\\_lock](#)
- ios\_base::openmode [\\_M\\_mode](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)
- char\_type \* [\\_M\\_out\\_end](#)
- bool [\\_M\\_reading](#)
- [\\_\\_state\\_type](#) [\\_M\\_state\\_beg](#)
- [\\_\\_state\\_type](#) [\\_M\\_state\\_cur](#)
- [\\_\\_state\\_type](#) [\\_M\\_state\\_last](#)

- `bool _M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

### 5.67.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 50 of file `stdio_filebuf.h`.

### 5.67.2 Constructor & Destructor Documentation

#### 5.67.2.1 `stdio_filebuf()` [1/3]

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf ( ) [inline]
```

deferred initialization

Definition at line 65 of file `stdio_filebuf.h`.

#### 5.67.2.2 `stdio_filebuf()` [2/3]

```
template<typename _CharT , typename _Traits >
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
    int __fd,
    std::ios_base::openmode __mode,
    size_t __size = static_cast<size_t>(BUFSIZ) )
```

**Parameters**

<code>__fd</code>	An open file descriptor.
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 137 of file `stdio_filebuf.h`.

**5.67.2.3 `stdio_filebuf()`** [3/3]

```
template<typename _CharT , typename _Traits >
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
    std::__c_file * __f,
    std::ios_base::openmode __mode,
    size_t __size = static_cast<size_t>(BUFSIZ) )
```

**Parameters**

<code>__f</code>	An open <code>FILE*</code> .
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars. Defaults to system's <code>BUFSIZ</code> .

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 153 of file `stdio_filebuf.h`.

**5.67.2.4 `~stdio_filebuf()`**

```
template<typename _CharT , typename _Traits >
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::~~stdio_filebuf ( ) [virtual]
```

Closes the external data stream if the file descriptor constructor was used.

Definition at line 132 of file `stdio_filebuf.h`.

**5.67.3 Member Function Documentation**

### 5.67.3.1 `_M_create_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_create_pback ( ) [inline], [protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 199 of file `fstream`.

### 5.67.3.2 `_M_destroy_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback ( ) throw ( ) [inline], [protected],
[inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 216 of file `fstream`.

### 5.67.3.3 `_M_set_buffer()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_set_buffer (
    streamsize __off ) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 443 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::close()`.

#### 5.67.3.4 close()

```
template<typename _CharT, typename _Traits >
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
( ) [inherited]
```

Closes the currently associated file.

##### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 213 of file `fstream.tcc`.

#### 5.67.3.5 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

#### 5.67.3.6 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

### 5.67.3.7 `epptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::epptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 542 of file `streambuf`.

### 5.67.3.8 `fd()`

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
int __gnu_cxx::stdio_filebuf<_CharT, _Traits>::fd ( ) [inline]
```

#### Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 118 of file `stdio_filebuf.h`.

### 5.67.3.9 `file()`

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
std::__c_file* __gnu_cxx::stdio_filebuf<_CharT, _Traits>::file ( ) [inline]
```

#### Returns

The underlying `FILE*`.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 128 of file `stdio_filebuf.h`.

### 5.67.3.10 `gbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::gbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the read position.



**Parameters**

<code>_↔ _n</code>	The delta by which to move.
------------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

**5.67.3.11 getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

**5.67.3.12 gptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

**5.67.3.13 imbue()**

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale &__loc __attribute__((unused)) ) [inline], [protected], [virtual],
[inherited]
```

Changes translations.

## Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Definition at line 583 of file `streambuf`.

5.67.3.14 `in_avail()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail ( ) [inline], [inherited]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

5.67.3.15 `is_open()`

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::is_open ( ) const throw ( ) [inline], [inherited]
```

Returns true if the external file is open.

Definition at line 260 of file `fstream`.

5.67.3.16 `open()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT, _Traits>::open (
    const char * __s,
    ios_base::openmode __mode ) [inherited]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

**Returns**

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

ios_base Flag combination					stdio equivalent
binary	in	out	trunc	app	
		+			w
		+		+	a
				+	a
		+	+		w
	+				r
	+	+			r+
	+	+	+		w+
	+	+		+	a+
	+			+	a+
+		+			wb
+		+		+	ab
+				+	ab
+		+	+		wb
+	+				rb
+	+	+			r+b
+	+	+	+		w+b
+	+	+		+	a+b
+	+			+	a+b

Definition at line 179 of file `fstream.tcc`.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

**5.67.3.17 open()** [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type* std::basic_filebuf< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode ) [inline], [inherited]
```

Opens an external file.

## Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

## Returns

`this` on success, `NULL` on failure

Definition at line 315 of file `fstream`.

5.67.3.18 `overflow()`

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::overflow (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

## Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

## Note

Base class version does nothing, returns `eof()`.

Definition at line 775 of file `streambuf`.

#### 5.67.3.19 pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
    int_type __c __attribute__((__unused__)) = traits_type::eof() )    [inline], [protected],
[virtual], [inherited]
```

Tries to back up the input sequence.

## Parameters

<code>_↔</code>	The character to be inserted back into the sequence.
<code>_C</code>	

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Definition at line 731 of file `streambuf`.

5.67.3.20 `pbase()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

5.67.3.21 `pbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

<code>_↔</code> <code>_n</code>	The delta by which to move.
------------------------------------	-----------------------------

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**5.67.3.22 pptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**5.67.3.23 pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc ) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

5.67.3.24 `pubseekoff()`

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf<_CharT, _Traits>::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekoff` function.

Definition at line 258 of file `streambuf`.

5.67.3.25 `pubseekpos()`

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf<_CharT, _Traits>::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 270 of file `streambuf`.

5.67.3.26 `pubsetbuf()`

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf<_CharT, _Traits>::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```



Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 5.67.3.27 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 5.67.3.28 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 5.67.3.29 `seekoff()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out ) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 798 of file `fstream.tcc`.

5.67.3.30 `seekpos()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out ) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 858 of file `fstream.tcc`.

5.67.3.31 `setbuf()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↵
::setbuf (
    char_type * __s,
    streamsize __n ) [protected], [virtual], [inherited]
```

Manipulates the buffer.

**Parameters**

<code>↵ __s</code>	Pointer to a buffer area.
<code>↵ __n</code>	Size of <code>__s</code> .

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↵html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 769 of file `fstream.tcc`.

### 5.67.3.32 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected], [inherited]
```

Setting the three read area pointers.

#### Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

#### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

### 5.67.3.33 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend ) [inline], [protected], [inherited]
```

Setting the three write area pointers.

#### Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

#### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

**5.67.3.34 `sgetc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

**5.67.3.35 `sgetn()`**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

**5.67.3.36 `showmanyc()`**

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf<_CharT, _Traits>::showmanyc ( ) [protected], [virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 263 of file `fstream.tcc`.

**5.67.3.37 `snextc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**5.67.3.38 `sputbackc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
    char_type __c ) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

<code>__c</code>	The character to push back.
------------------	-----------------------------

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**5.67.3.39 `sputc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
    char_type __c ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

<code>__c</code>	A character to output.
------------------	------------------------

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.67.3.40 `sputn()`**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

$\_s$	A buffer read area.
$\_n$	A count.

One of two public output functions.

Returns `xspn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**5.67.3.41 `sungetc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**5.67.3.42 `sync()`**

```
template<typename _CharT , typename _Traits >
int std::basic_filebuf< _CharT, _Traits >::sync ( ) [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 978 of file `fstream.tcc`.

5.67.3.43 `uflow()`

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::uflow ( ) [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 707 of file `streambuf`.

5.67.3.44 `underflow()`

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::underflow ( )
[protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 289 of file `fstream.tcc`.

5.67.3.45 `xsgetn()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::xsgetn (
    char_type * __s,
    streamsize __n ) [protected], [virtual], [inherited]
```

Multiple character extraction.



**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 635 of file `fstream.tcc`.

**5.67.3.46 xspn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::xspn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 721 of file `fstream.tcc`.

#### 5.67.4 Member Data Documentation

##### 5.67.4.1 `_M_buf`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf [protected], [inherited]
```

Pointer to the beginning of internal buffer.

Definition at line 136 of file `fstream`.

##### 5.67.4.2 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file `streambuf`.

##### 5.67.4.3 `_M_buf_size`

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf<_CharT, _Traits>::_M_buf_size [protected], [inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 143 of file `fstream`.

##### 5.67.4.4 `_M_ext_buf`

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf<_CharT, _Traits>::_M_ext_buf [protected], [inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 178 of file `fstream`.

#### 5.67.4.5 `_M_ext_buf_size`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected], [inherited]
```

Size of buffer held by `_M_ext_buf`.

Definition at line 183 of file `fstream`.

#### 5.67.4.6 `_M_ext_next`

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected], [inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 190 of file `fstream`.

#### 5.67.4.7 `_M_in_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file `streambuf`.

#### 5.67.4.8 `_M_in_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file `streambuf`.

#### 5.67.4.9 `_M_in_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file `streambuf`.

#### 5.67.4.10 `_M_mode`

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf<_CharT, _Traits>::_M_mode [protected], [inherited]
```

Place to stash in || out || in | out settings for current filebuf.

Definition at line 121 of file fstream.

Referenced by `std::basic_filebuf<char_type, traits_type>::close()`.

#### 5.67.4.11 `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 5.67.4.12 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 5.67.4.13 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file streambuf.

#### 5.67.4.14 `_M_pback`

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 164 of file fstream.

#### 5.67.4.15 `_M_pback_cur_save`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 165 of file fstream.

#### 5.67.4.16 `_M_pback_end_save`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 166 of file fstream.

## 5.67.4.17 \_M\_pback\_init

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::_M_pback_init [protected], [inherited]
```

Necessary bits for putback buffer management.

## Note

pbacks of over one character are not currently supported.

Definition at line 167 of file fstream.

Referenced by `std::basic_filebuf<char_type, traits_type>::close()`.

## 5.67.4.18 \_M\_reading

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::_M_reading [protected], [inherited]
```

\_M\_reading == false && \_M\_writing == false for **uncommitted** mode; \_M\_reading == true for **read** mode; \_M\_writing == true for **write** mode;

NB: \_M\_reading == true && \_M\_writing == true is unused.

Definition at line 155 of file fstream.

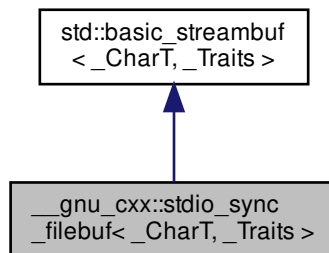
Referenced by `std::basic_filebuf<char_type, traits_type>::close()`.

The documentation for this class was generated from the following file:

- [stdio\\_filebuf.h](#)

## 5.68 \_\_gnu\_cxx::stdio\_sync\_filebuf&lt;\_CharT, \_Traits&gt; Class Template Reference

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `_Traits` **traits\_type**

## Public Member Functions

- **stdio\_sync\_filebuf** (`std::__c_file * __f`)
  - **stdio\_sync\_filebuf** (`stdio_sync_filebuf && __fb`) noexcept
  - `std::__c_file * file` ()
  - locale `getloc` () const
  - streamsize `in_avail` ()
  - `stdio_sync_filebuf & operator=` (`stdio_sync_filebuf && __fb`) noexcept
  - locale `pubimbue` (const locale & \_\_loc)
  - `int_type sbumpc` ()
  - `int_type sgetc` ()
  - streamsize `sgetn` (`char_type * __s`, streamsize \_\_n)
  - `int_type snextc` ()
  - `int_type sputbackc` (`char_type __c`)
  - `int_type sputc` (`char_type __c`)
  - streamsize `sputn` (const `char_type * __s`, streamsize \_\_n)
  - `int_type sungetc` ()
  - void **swap** (`stdio_sync_filebuf & __fb`)
- 
- `basic_streambuf * pubsetbuf` (`char_type * __s`, streamsize \_\_n)
  - `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - int `pubsync` ()

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
- void **\_\_safe\_pbump** (streamsize \_\_n)
- void `gbump` (int \_\_n)
- virtual void `imbue` (const locale & \_\_loc \_\_attribute\_\_((\_\_unused\_\_)))
- virtual `int_type overflow` (`int_type __c=traits_type::eof()`)
- virtual `int_type overflow` (`int_type __c __attribute__((__unused__))=traits_type::eof()`)
- virtual `int_type pbackfail` (`int_type __c=traits_type::eof()`)
- virtual `int_type pbackfail` (`int_type __c __attribute__((__unused__))=traits_type::eof()`)
- void `pbump` (int \_\_n)
- virtual `std::streampos seekoff` (`std::streamoff __off`, `std::ios_base::seekdir __dir`, `std::ios_base::openmode=std::ios_base::in|std::ios_base::out`)
- virtual `pos_type seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode=ios_base::in|ios_base::out`)

- virtual `std::streampos seekpos (std::streampos __pos, std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out)`
  - virtual `pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)`
  - virtual `basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)`
  - void `setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)`
  - void `setp (char_type * __pbeg, char_type * __pend)`
  - virtual streamsize `showmanyc ()`
  - void `swap (basic_streambuf & __sb)`
  - virtual int `sync ()`
  - `int_type syncgetc ()`
  - template<>  
`stdio_sync_filebuf< char >::int_type syncgetc ()`
  - template<>  
`stdio_sync_filebuf< wchar_t >::int_type syncgetc ()`
  - `int_type syncputc (int_type __c)`
  - template<>  
`stdio_sync_filebuf< char >::int_type syncputc (int_type __c)`
  - template<>  
`stdio_sync_filebuf< wchar_t >::int_type syncputc (int_type __c)`
  - `int_type syncungetc (int_type __c)`
  - template<>  
`stdio_sync_filebuf< char >::int_type syncungetc (int_type __c)`
  - template<>  
`stdio_sync_filebuf< wchar_t >::int_type syncungetc (int_type __c)`
  - virtual `int_type uflow ()`
  - virtual `int_type underflow ()`
  - virtual `std::streamsize xsgetn (char_type * __s, std::streamsize __n)`
  - template<>  
`std::streamsize xsgetn (char * __s, std::streamsize __n)`
  - template<>  
`std::streamsize xsgetn (wchar_t * __s, std::streamsize __n)`
  - virtual streamsize `xsgetn (char_type * __s, streamsize __n)`
  - virtual `std::streamsize xspu (const char_type * __s, std::streamsize __n)`
  - template<>  
`std::streamsize xspu (const char * __s, std::streamsize __n)`
  - template<>  
`std::streamsize xspu (const wchar_t * __s, std::streamsize __n)`
  - virtual streamsize `xspu (const char_type * __s, streamsize __n)`
- 
- `char_type * eback () const`
  - `char_type * gptr () const`
  - `char_type * egptr () const`
- 
- `char_type * pbase () const`
  - `char_type * pptr () const`
  - `char_type * ep_ptr () const`



### Protected Attributes

- locale [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)

### 5.68.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 57 of file `stdio_sync_filebuf.h`.

### 5.68.2 Member Function Documentation

#### 5.68.2.1 `eback()`

```
template<typename _CharT, typename _Traits>
char\_type* std::basic\_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

### 5.68.2.2 `egptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::egptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

### 5.68.2.3 `epptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::epptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 542 of file `streambuf`.

### 5.68.2.4 `file()`

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file* __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file ( ) [inline]
```

#### Returns

The underlying `FILE*`.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 118 of file `stdio_sync_filebuf.h`.

### 5.68.2.5 `gbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::gbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

<code>_↔ _n</code>	The delta by which to move.
------------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

**5.68.2.6 getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

**5.68.2.7 gptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

**5.68.2.8 imbue()**

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale &__loc __attribute__((unused)) ) [inline], [protected], [virtual],
[inherited]
```

Changes translations.

## Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Definition at line 583 of file `streambuf`.

5.68.2.9 `in_avail()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail ( ) [inline], [inherited]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

5.68.2.10 `overflow()`

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::overflow (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 775 of file streambuf.

**5.68.2.11 pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits >::pbackfail (
    int_type __c __attribute__((__unused__)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

<code>__c</code>	The character to be inserted back into the sequence.
------------------	--

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 731 of file streambuf.

#### 5.68.2.12 `pbase()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

#### 5.68.2.13 `pbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the write position.

##### Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the write position without returning any data.

Definition at line 552 of file `streambuf`.

#### 5.68.2.14 `pptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 539 of file `streambuf`.

#### 5.68.2.15 pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc ) [inline], [inherited]
```

Entry point for imbue().

##### Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

##### Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

#### 5.68.2.16 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

##### Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for ios_base::seekdir.
<code>__mode</code>	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

#### 5.68.2.17 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
```

```
pos_type __sp,  
ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.



**Parameters**

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 270 of file `streambuf`.

**5.68.2.18 `pubsetbuf()`**

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

**5.68.2.19 `pubsync()`**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

**5.68.2.20 `sbumpc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 5.68.2.21 `seekoff()`

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 609 of file `streambuf`.

#### 5.68.2.22 `seekpos()`

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 621 of file `streambuf`.

#### 5.68.2.23 setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf<char_type,_Traits>* std::basic_streambuf< _CharT, _Traits >::setbuf (
    char_type * ,
    streamsize ) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

##### Note

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 598 of file `streambuf`.

#### 5.68.2.24 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected], [inherited]
```

Setting the three read area pointers.

##### Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

##### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file `streambuf`.

5.68.2.25 `setp()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
    char_type * __pbeg,
    char_type * __pend ) [inline], [protected], [inherited]
```

Setting the three write area pointers.

## Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

## Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file `streambuf`.

5.68.2.26 `sgetc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc ( ) [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

5.68.2.27 `sgetn()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

$\_s$	A buffer area.
$\_n$	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

**5.68.2.28 showmanyc()**

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ( ) [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 656 of file `streambuf`.

#### 5.68.2.29 snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

#### 5.68.2.30 sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
    char_type __c ) [inline], [inherited]
```

Pushing characters back into the input stream.

##### Parameters

<code>__c</code>	The character to push back.
------------------	-----------------------------

##### Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

#### 5.68.2.31 sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
    char_type __c ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

$\_c$	A character to output.
-------	------------------------

**Returns**

$\_c$ , if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores  $\_c$  in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵  
__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.68.2.32 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

$\_s$	A buffer read area.
$\_n$	A count.

One of two public output functions.

Returns `xputn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**5.68.2.33 sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**5.68.2.34 `sync()`**

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::sync (
    void ) [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 190 of file `stdio_sync_filebuf.h`.

**5.68.2.35 `uflow()`**

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow ( ) [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 138 of file `stdio_sync_filebuf.h`.



#### 5.68.2.36 underflow()

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::underflow ( ) [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 131 of file `stdio_sync_filebuf.h`.

#### 5.68.2.37 xsgetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
    char_type * __s,
    streamsize __n ) [protected], [virtual], [inherited]
```

Multiple character extraction.

##### Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 46 of file `streambuf.tcc`.

**5.68.2.38 xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 80 of file `streambuf.tcc`.

**5.68.3 Member Data Documentation**

#### 5.68.3.1 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file streambuf.

#### 5.68.3.2 `_M_in_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 5.68.3.3 `_M_in_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

#### 5.68.3.4 `_M_in_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 5.68.3.5 `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

5.68.3.6 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file `streambuf`.

5.68.3.7 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

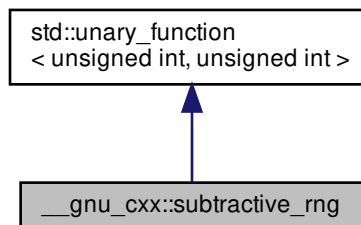
Definition at line 196 of file `streambuf`.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

5.69 `__gnu_cxx::subtractive_rng` Class Reference

Inheritance diagram for `__gnu_cxx::subtractive_rng`:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## Public Member Functions

- [subtractive\\_rng](#) (unsigned int \_\_seed)
- [subtractive\\_rng](#) ()
- void **\_M\_initialize** (unsigned int \_\_seed)
- unsigned int [operator\(\)](#) (unsigned int \_\_limit)

### 5.69.1 Detailed Description

The `subtractive_rng` class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

Definition at line 352 of file `ext/functional`.

### 5.69.2 Member Typedef Documentation

#### 5.69.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary\_function< _Arg, _Result >::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.69.2.2 `result_type`

```
template<typename _Arg, typename _Result>
typedef _Result std::unary\_function< _Arg, _Result >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

### 5.69.3 Constructor & Destructor Documentation

#### 5.69.3.1 `subtractive_rng()` [1/2]

```
\_\_gnu\_cxx::subtractive\_rng::subtractive\_rng (
    unsigned int __seed ) [inline]
```

Ctor allowing you to initialize the seed.

Definition at line 394 of file `ext/functional`.

5.69.3.2 `subtractive_rng()` [2/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng ( ) [inline]
```

Default ctor; initializes its state with some number you don't see.

Definition at line 398 of file `ext/functional`.

## 5.69.4 Member Function Documentation

5.69.4.1 `operator()()`

```
unsigned int __gnu_cxx::subtractive_rng::operator() (
    unsigned int __limit ) [inline]
```

Returns a number less than the argument.

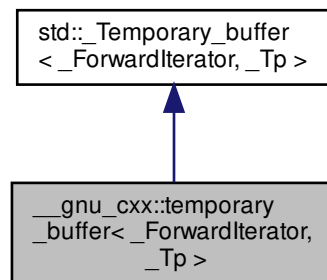
Definition at line 363 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

5.70 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- [temporary\\_buffer](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- [~temporary\\_buffer](#) ()
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

### Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

#### 5.70.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >
```

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

Definition at line 183 of file `ext/memory`.

#### 5.70.2 Constructor & Destructor Documentation

### 5.70.2.1 `temporary_buffer()`

```
template<class _ForwardIterator , class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Requests storage large enough to hold a copy of [first,last).

Definition at line 186 of file `ext/memory`.

### 5.70.2.2 `~temporary_buffer()`

```
template<class _ForwardIterator , class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::~~temporary_buffer ( ) [inline]
```

Destroys objects and frees storage.

Definition at line 190 of file `ext/memory`.

## 5.70.3 Member Function Documentation

### 5.70.3.1 `begin()`

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( ) [inline], [inherited]
```

As per Table mumble.

Definition at line 151 of file `stl_tempbuf.h`.

### 5.70.3.2 `end()`

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( ) [inline], [inherited]
```

As per Table mumble.

Definition at line 156 of file `stl_tempbuf.h`.



### 5.70.3.3 requested\_size()

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size ( ) const [inline],
[inherited]
```

Returns the size requested by the constructor; may be >size().

Definition at line 146 of file stl\_tempbuf.h.

### 5.70.3.4 size()

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const [inline], [inherited]
```

As per Table mumble.

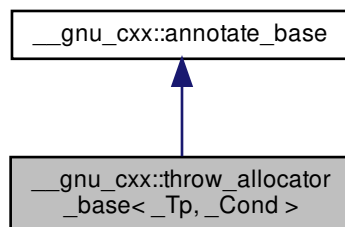
Definition at line 141 of file stl\_tempbuf.h.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

## 5.71 \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >:



### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check** (size\_type \_\_n)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_constructed** (void \*p)
- void **check\_constructed** (size\_t label)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)
- size\_type **max\_size** () const noexcept

## Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

## 5.71.1 Detailed Description

```
template<typename _Tp, typename _Cond>
class __gnu_cxx::throw_allocator_base<_Tp, _Cond>
```

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

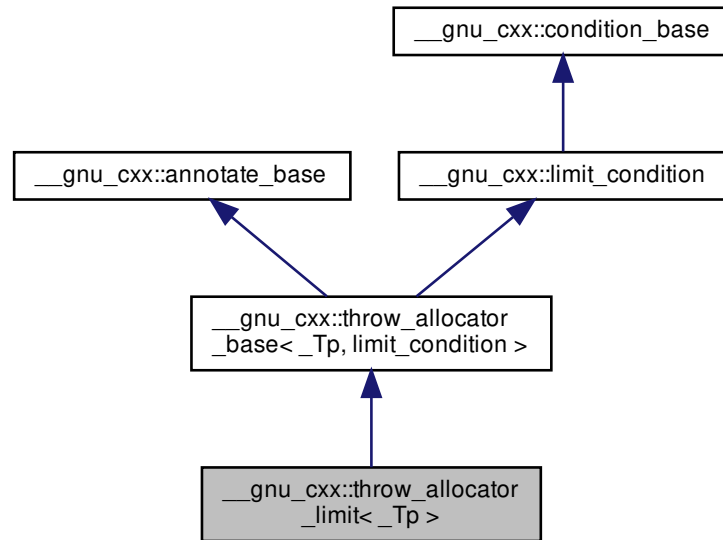
Definition at line 790 of file `throw_allocator.h`.

The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

## 5.72 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#) &) noexcept
- template<typename \_Tp1 >  
  **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#)<\_Tp1> &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check** (size\_type \_\_n)
- void **check\_allocated** (void \*p, size\_t size)

- void **check\_allocated** (pointer `__p`, size\_type `__n`)
- void **check\_constructed** (void `*p`)
- void **check\_constructed** (size\_t `label`)
- void **construct** (`_Up *``__p`, `_Args &&...` `__args`)
- void **deallocate** (pointer `__p`, size\_type `__n`)
- void **destroy** (`_Up *``__p`)
- void **erase** (void `*p`, size\_t `size`)
- void **erase\_construct** (void `*p`)
- void **insert** (void `*p`, size\_t `size`)
- void **insert\_construct** (void `*p`)
- size\_type **max\_size** () const noexcept

#### Static Public Member Functions

- static void **check** ()
- static size\_t & **count** ()
- static size\_t **get\_label** ()
- static size\_t & **limit** ()
- static void **set\_label** (size\_t `l`)
- static void **set\_limit** (const size\_t `__l`)
- static void **throw\_conditionally** ()

#### 5.72.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit<_Tp>
```

Allocator throwing via limit condition.

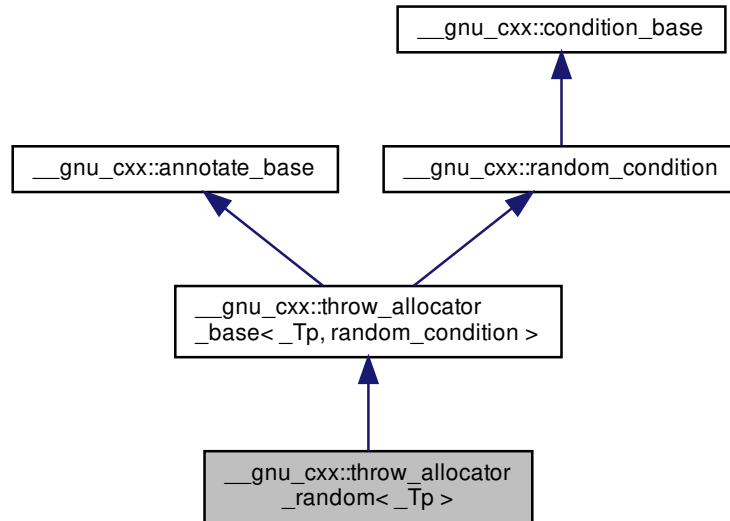
Definition at line 899 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.73 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



#### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) noexcept
- template<typename \_Tp1 >  
**throw\_allocator\_random** (const [throw\\_allocator\\_random](#)<\_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check** (size\_type \_\_n)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)

- void **check\_constructed** (void \*p)
- void **check\_constructed** (size\_t label)
- void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)
- size\_type **max\_size** () const noexcept
- void **seed** (unsigned long \_\_s)

#### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)
- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

#### 5.73.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_random<_Tp>
```

Allocator throwing via random condition.

Definition at line 920 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.74 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Inherits `_Cond`.

#### Public Types

- typedef `_Cond` **condition\_type**

### Public Member Functions

- **throw\_value\_base** (const [throw\\_value\\_base](#) &\_\_v)
- **throw\_value\_base** ([throw\\_value\\_base](#) &&)=default
- **throw\_value\_base** (const std::size\_t \_\_i)
- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_base](#) & **operator=** (const [throw\\_value\\_base](#) &\_\_v)
- [throw\\_value\\_base](#) & **operator=** ([throw\\_value\\_base](#) &&)=default

### Public Attributes

- std::size\_t **M\_i**

#### 5.74.1 Detailed Description

```
template<typename _Cond>
struct __gnu_cxx::throw_value_base< _Cond >
```

Class with exception generation control. Intended to be used as a value\_type in templated code.

Note: Destructor not allowed to throw.

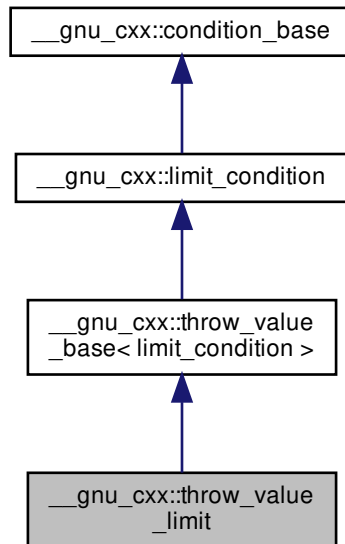
Definition at line 603 of file throw\_allocator.h.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.75 `__gnu_cxx::throw_value_limit` Struct Reference

Inheritance diagram for `__gnu_cxx::throw_value_limit`:



## Public Types

- typedef `throw_value_base< limit_condition >` **base\_type**
- typedef `limit_condition` **condition\_type**

## Public Member Functions

- **throw\_value\_limit** (const `throw_value_limit` &\_\_other)
- **throw\_value\_limit** (`throw_value_limit` &&)=default
- **throw\_value\_limit** (const std::size\_t \_\_i)
- `throw_value_base` & **operator++** ()
- `throw_value_limit` & **operator=** (const `throw_value_limit` &\_\_other)
- `throw_value_limit` & **operator=** (`throw_value_limit` &&)=default

## Static Public Member Functions

- static size\_t & **count** ()
- static size\_t & **limit** ()
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()



#### Public Attributes

- `std::size_t _M_i`

#### 5.75.1 Detailed Description

Type throwing via limit condition.

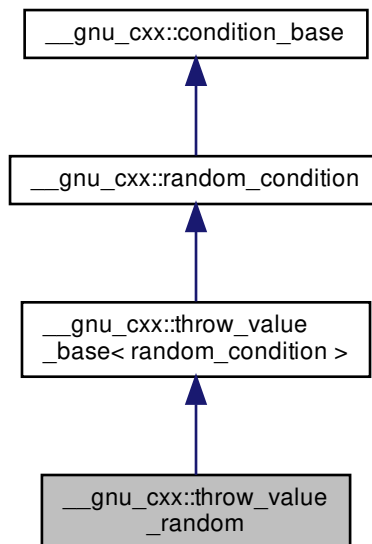
Definition at line 720 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 5.76 `__gnu_cxx::throw_value_random` Struct Reference

Inheritance diagram for `__gnu_cxx::throw_value_random`:



#### Public Types

- typedef [throw\\_value\\_base< random\\_condition >](#) **base\_type**
- typedef [random\\_condition](#) **condition\_type**

## Public Member Functions

- `throw_value_random` (const [throw\\_value\\_random](#) &\_\_other)
- `throw_value_random` ([throw\\_value\\_random](#) &&)=default
- `throw_value_random` (const std::size\_t \_\_i)
- [throw\\_value\\_base](#) & `operator++` ()
- [throw\\_value\\_random](#) & `operator=` (const [throw\\_value\\_random](#) &\_\_other)
- [throw\\_value\\_random](#) & `operator=` ([throw\\_value\\_random](#) &&)=default
- void `seed` (unsigned long \_\_s)

## Static Public Member Functions

- static void `set_probability` (double \_\_p)
- static void `throw_conditionally` ()

## Public Attributes

- std::size\_t `M_i`

## 5.76.1 Detailed Description

Type throwing via random condition.

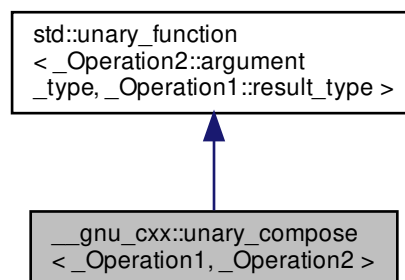
Definition at line 751 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.77 `__gnu_cxx::unary_compose< _Operation1, _Operation2 >` Class Template Reference

Inheritance diagram for `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- **unary\_compose** (const `_Operation1` &\_\_x, const `_Operation2` &\_\_y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &\_\_x) const

### Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**

#### 5.77.1 Detailed Description

```
template<class _Operation1, class _Operation2>
class __gnu_cxx::unary_compose< _Operation1, _Operation2 >
```

An [SGI extension](#) .

Definition at line 125 of file `ext/functional`.

#### 5.77.2 Member Typedef Documentation

##### 5.77.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary\_function< _Arg, _Result >::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

##### 5.77.2.2 `result_type`

```
template<typename _Arg, typename _Result>
typedef _Result std::unary\_function< _Arg, _Result >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.78 `__gnu_debug::After_nth_from<_Iterator>` Class Template Reference

### Public Member Functions

- **`After_nth_from`** (const difference\_type &\_\_n, const \_Iterator &\_\_base)
- bool **`operator()`** (const \_Iterator &\_\_x) const

#### 5.78.1 Detailed Description

```
template<typename _Iterator>
class __gnu_debug::After_nth_from<_Iterator>
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 74 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.79 `__gnu_debug::BeforeBeginHelper<_Sequence>` Struct Template Reference

### Static Public Member Functions

- template<typename \_Iterator>  
static bool **`_S_Is`** (const [Safe\\_iterator](#)<\_Iterator, \_Sequence> &)
- template<typename \_Iterator>  
static bool **`_S_Is_Beginnest`** (const [Safe\\_iterator](#)<\_Iterator, \_Sequence> &\_\_it)

#### 5.79.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::BeforeBeginHelper<_Sequence>
```

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 45 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 5.80 `__gnu_debug::Equal_to<_Type>` Class Template Reference

### Public Member Functions

- `Equal_to` (const `_Type` &\_\_v)
- bool `operator()` (const `_Type` &\_\_x) const

#### 5.80.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::Equal_to<_Type>
```

A simple function object that returns true if the passed-in value is equal to the stored value.

Definition at line 59 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.81 `__gnu_debug::Not_equal_to<_Type>` Class Template Reference

### Public Member Functions

- `Not_equal_to` (const `_Type` &\_\_v)
- bool `operator()` (const `_Type` &\_\_x) const

#### 5.81.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::Not_equal_to<_Type>
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

Definition at line 44 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.82 `__gnu_debug::Safe_container<_SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware>` Class Template Reference

Inherits `_SafeBase<_SafeContainer>`.

## Public Member Functions

- void **\_M\_swap** ([\\_Safe\\_container](#) &\_\_x) noexcept
- [\\_Safe\\_container](#) & **operator=** (const [\\_Safe\\_container](#) &) noexcept
- [\\_Safe\\_container](#) & **operator=** ([\\_Safe\\_container](#) &&\_\_x) noexcept

## Protected Member Functions

- **\_Safe\_container** (const [\\_Safe\\_container](#) &)=default
- **\_Safe\_container** ([\\_Safe\\_container](#) &&)=default
- **\_Safe\_container** ([\\_Safe\\_container](#) &&\_\_x, const [\\_Alloc](#) &\_\_a)
- [\\_Safe\\_container](#) & **\_M\_safe** () noexcept

## 5.82.1 Detailed Description

```
template<typename _SafeContainer, typename _Alloc, template< typename > class _SafeBase, bool _IsCxx11AllocatorAware = true>
class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >
```

Safe class dealing with some allocator dependent operations.

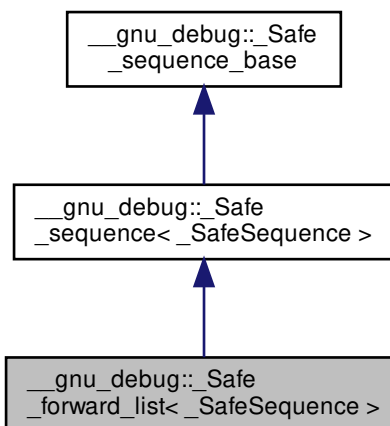
Definition at line 41 of file `safe_container.h`.

The documentation for this class was generated from the following file:

- [safe\\_container.h](#)

## 5.83 \_\_gnu\_debug::\_Safe\_forward\_list&lt;\_SafeSequence&gt; Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_forward_list<_SafeSequence>`:



### Public Member Functions

- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) ([\\_Safe\\_sequence](#) &\_\_from, [\\_Predicate](#) \_\_pred)

### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &) noexcept

#### 5.83.1 Detailed Description

```
template<typename _SafeSequence>
class __gnu_debug::_Safe_forward_list< _SafeSequence >
```

Special iterators swap and invalidation for `forward_list` because of the `before_begin` iterator.

Definition at line 51 of file `debug/forward_list`.

#### 5.83.2 Member Function Documentation

##### 5.83.2.1 [\\_M\\_detach\\_all\(\)](#)

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

5.83.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

**Postcondition**

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.83.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

5.83.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

5.83.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence<_SafeSequence>::_M_invalidate_if (   
    _Predicate __pred ) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.



### 5.83.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### 5.83.2.7 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_transfer_from_if (
    _Safe_sequence< _SafeSequence > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

## 5.83.3 Member Data Documentation

### 5.83.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.83.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.83.3.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file safe\_base.h.

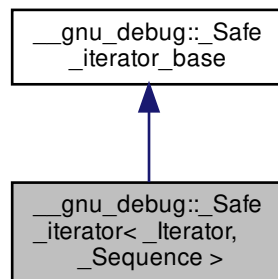
Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_invalidate\_all().

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

## 5.84 \_\_gnu\_debug::\_Safe\_iterator&lt;\_Iterator, \_Sequence &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence >:



## Public Types

- typedef \_\_Traits::difference\_type **difference\_type**
- typedef \_\_Traits::iterator\_category **iterator\_category**
- typedef \_\_Iterator **iterator\_type**
- typedef \_\_Traits::pointer **pointer**
- typedef \_\_Traits::reference **reference**
- typedef \_\_Traits::value\_type **value\_type**

## Public Member Functions

- [\\_Safe\\_iterator](#) () noexcept
- [\\_Safe\\_iterator](#) (const [\\_Iterator](#) &\_\_i, const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) noexcept
- [\\_Safe\\_iterator](#) (const [\\_Safe\\_iterator](#) &\_\_x) noexcept
- [\\_Safe\\_iterator](#) ([\\_Safe\\_iterator](#) &&\_\_x) noexcept
- template<typename [\\_MutableIterator](#) >  
[\\_Safe\\_iterator](#) (const [\\_Safe\\_iterator](#)< [\\_MutableIterator](#), typename [\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)<(std::\_\_are\_same<  
[\\_MutableIterator](#), typename [\\_Sequence::iterator::iterator\\_type](#) >::\_\_value), [\\_Sequence](#) >::\_\_type > &\_\_x) noex-  
cept
- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq)
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_before\\_dereferenceable](#) () const
- bool [\\_M\\_can\\_advance](#) (const difference\_type &\_\_n) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- bool [\\_M\\_decrementable](#) () const
- bool [\\_M\\_dereferenceable](#) () const
- void [\\_M\\_detach\\_single](#) () throw ()
- [\\_\\_gnu\\_cxx::\\_\\_conditional\\_type](#)< std::\_\_are\_same< [\\_Const\\_iterator](#), [\\_Safe\\_iterator](#) >::\_\_value, const [\\_↔](#)  
[Sequence](#) \*, [\\_Sequence](#) \* >::\_\_type [\\_M\\_get\\_sequence](#) () const
- bool [\\_M\\_incrementable](#) () const
- void [\\_M\\_invalidate](#) ()
- bool [\\_M\\_is\\_before\\_begin](#) () const
- bool [\\_M\\_is\\_begin](#) () const
- bool [\\_M\\_is\\_beginnest](#) () const
- bool [\\_M\\_is\\_end](#) () const
- void [\\_M\\_reset](#) () throw ()
- bool [\\_M\\_singular](#) () const throw ()
- void [\\_M\\_unlink](#) () throw ()
- bool [\\_M\\_valid\\_range](#) (const [\\_Safe\\_iterator](#) &\_\_rhs, [std::pair](#)< difference\_type, [\\_Distance\\_precision](#) > &\_\_dist,  
bool [\\_\\_check\\_dereferenceable](#)=true) const
- [\\_Iterator](#) & [base](#) () noexcept
- const [\\_Iterator](#) & [base](#) () const noexcept
- [operator \\_Iterator](#) () const noexcept
- reference [operator\\*](#) () const noexcept
- [\\_Safe\\_iterator](#) [operator+](#) (const difference\_type &\_\_n) const noexcept
- [\\_Safe\\_iterator](#) & [operator++](#) () noexcept
- [\\_Safe\\_iterator](#) [operator++](#) (int) noexcept
- [\\_Safe\\_iterator](#) & [operator+=](#) (const difference\_type &\_\_n) noexcept
- [\\_Safe\\_iterator](#) [operator-](#) (const difference\_type &\_\_n) const noexcept
- [\\_Safe\\_iterator](#) & [operator--](#) () noexcept
- [\\_Safe\\_iterator](#) [operator--](#) (int) noexcept
- [\\_Safe\\_iterator](#) & [operator-=](#) (const difference\_type &\_\_n) noexcept
- pointer [operator->](#) () const noexcept
- [\\_Safe\\_iterator](#) & [operator=](#) (const [\\_Safe\\_iterator](#) &\_\_x) noexcept
- [\\_Safe\\_iterator](#) & [operator=](#) ([\\_Safe\\_iterator](#) &&\_\_x) noexcept
- reference [operator\[\]](#) (const difference\_type &\_\_n) const noexcept

**Public Attributes**

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

**Protected Member Functions**

- `void _M_attach (_Safe_sequence_base * __seq, bool __constant)`
- `void _M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()`
- `void _M_detach ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

**5.84.1 Detailed Description**

```
template<typename _Iterator, typename _Sequence>
class __gnu_debug::_Safe_iterator<_Iterator, _Sequence>
```

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Note that `_Iterator` must be the first base class so that it gets initialized before the iterator is being attached to the container's list of iterators and it is being detached before `_Iterator` get destroyed. Otherwise it would result in a data race.

Definition at line 56 of file `formatter.h`.

**5.84.2 Constructor & Destructor Documentation****5.84.2.1 `_Safe_iterator()` [1/5]**

```
template<typename _Iterator, typename _Sequence>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator ( ) [inline], [noexcept]
```

**Postcondition**

the iterator is singular and unattached

Definition at line 119 of file `safe_iterator.h`.

**5.84.2.2** `_Safe_iterator()` [2/5]

```
template<typename _Iterator, typename _Sequence>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator (
    const _Iterator & __i,
    const _Safe_sequence_base * __seq ) [inline], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.

**Precondition**

`seq` is not NULL

**Postcondition**

this is not singular

Definition at line 128 of file `safe_iterator.h`.

**5.84.2.3** `_Safe_iterator()` [3/5]

```
template<typename _Iterator, typename _Sequence>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator (
    const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline], [noexcept]
```

Copy construction.

Definition at line 140 of file `safe_iterator.h`.

**5.84.2.4** `_Safe_iterator()` [4/5]

```
template<typename _Iterator, typename _Sequence>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator (
    _Safe_iterator< _Iterator, _Sequence > && __x ) [inline], [noexcept]
```

Move construction.

**Postcondition**

`__x` is singular and unattached

Definition at line 158 of file `safe_iterator.h`.

## 5.84.2.5 \_Safe\_iterator() [5/5]

```
template<typename _Iterator, typename _Sequence>
template<typename _MutableIterator >
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator (
    const _Safe_iterator< _MutableIterator, typename __gnu_cxx::__enable_if<(std::__is_same<
    _MutableIterator, typename _Sequence::iterator::iterator_type >::__value), _Sequence
    >::__type > & __x ) [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 178 of file `safe_iterator.h`.

## 5.84.3 Member Function Documentation

## 5.84.3.1 \_M\_attach() [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
    _Safe_sequence_base * __seq,
    bool __constant ) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_attach()`, and `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`.

## 5.84.3.2 \_M\_attach() [2/2]

```
template<typename _Iterator, typename _Sequence>
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach (
    _Safe_sequence_base * __seq ) [inline]
```

Attach iterator to the given sequence.

Definition at line 416 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_attach()`.

#### 5.84.3.3 `_M_attach_single()` [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
    _Safe_sequence_base * __seq,
    bool __constant ) throw ( )    [protected], [inherited]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach_single()`.

#### 5.84.3.4 `_M_attach_single()` [2/2]

```
template<typename _Iterator, typename _Sequence>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach_single (
    _Safe_sequence_base * __seq )    [inline]
```

Likewise, but not thread-safe.

Definition at line 421 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_attach_single()`.

#### 5.84.3.5 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const _Safe_sequence_base * __seq ) const    [inline], [inherited]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

#### 5.84.3.6 `_M_before_dereferenceable()`

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_before_dereferenceable ( ) const
[inline]
```

Is the iterator before a dereferenceable one?

Definition at line 431 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::base()`.

### 5.84.3.7 \_M\_can\_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const __Safe_iterator_base & __x ) const throw ( )    [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

### 5.84.3.8 \_M\_dereferenceable()

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable ( ) const [inline]
```

Is the iterator dereferenceable?

Definition at line 426 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_before_begin()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_check_dereferenceable()`.

### 5.84.3.9 \_M\_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach ( ) [protected], [inherited]
```

Detach the iterator for whatever sequence it is attached to, if any.

### 5.84.3.10 \_M\_detach\_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single ( ) throw ( )    [inherited]
```

Likewise, but not thread-safe.

### 5.84.3.11 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw ( )    [protected],
[inherited]
```

For use in `_Safe_iterator`.

### 5.84.3.12 \_M\_incrementable()

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable ( ) const [inline]
```

Is the iterator incrementable?

Definition at line 443 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_before_dereferenceable()`.



#### 5.84.3.13 `_M_invalidate()`

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_version`.

#### 5.84.3.14 `_M_is_before_begin()`

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_before_begin ( ) const [inline]
```

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Definition at line 482 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

#### 5.84.3.15 `_M_is_begin()`

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin ( ) const [inline]
```

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 471 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base()`.

#### 5.84.3.16 `_M_is_beginnest()`

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_beginnest ( ) const [inline]
```

Is this iterator equal to the sequence's `before_begin()` iterator if any or `begin()` otherwise?

Definition at line 488 of file `safe_iterator.h`.

## 5.84.3.17 \_M\_is\_end()

```
template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_end ( ) const [inline]
```

Is this iterator equal to the sequence's end() iterator?

Definition at line 476 of file safe\_iterator.h.

References \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::base().

Referenced by \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_dereferenceable(), and \_\_gnu\_debug::\_↵  
Safe\_iterator<\_Iterator, \_Sequence>::\_M\_incrementable().

## 5.84.3.18 \_M\_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw ( ) [inherited]
```

Reset all member variables

## 5.84.3.19 \_M\_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ( ) [inherited]
```

Is this iterator singular?

Referenced by \_\_gnu\_debug::\_check\_singular\_aux(), \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_dereferenceable(), \_\_gnu↵  
\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_incrementable(), and \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_incrementable().

## 5.84.3.20 \_M\_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw ( ) [inline], [inherited]
```

Unlink itself

Definition at line 155 of file safe\_base.h.

#### 5.84.3.21 base()

```
template<typename _Iterator, typename _Sequence>
_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base ( ) [inline], [noexcept]
```

Return the underlying iterator.

Definition at line 403 of file safe\_iterator.h.

Referenced by `__gnu_debug::_get_distance()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::M_before_↵  
dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::M_is_begin()`, and `__gnu_debug::_Safe_↵  
_iterator< _Iterator, _Sequence >::M_is_end()`.

#### 5.84.3.22 operator\_iterator()

```
template<typename _Iterator, typename _Sequence>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator_iterator ( ) const [inline],
[noexcept]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 412 of file safe\_iterator.h.

#### 5.84.3.23 operator\*()

```
template<typename _Iterator, typename _Sequence>
reference __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator* ( ) const [inline],
[noexcept]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 266 of file safe\_iterator.h.

#### 5.84.3.24 operator++() [1/2]

```
template<typename _Iterator, typename _Sequence>
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++ ( ) [inline],
[noexcept]
```

Iterator preincrement.

##### Precondition

iterator is incrementable

Definition at line 293 of file safe\_iterator.h.

#### 5.84.3.25 operator++() [2/2]

```
template<typename _Iterator, typename _Sequence>
__Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++ (
    int ) [inline], [noexcept]
```

Iterator postincrement.

##### Precondition

iterator is incrementable

Definition at line 308 of file safe\_iterator.h.

#### 5.84.3.26 operator--() [1/2]

```
template<typename _Iterator, typename _Sequence>
__Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- ( ) [inline],
[noexcept]
```

Iterator predecrement.

##### Precondition

iterator is decrementable

Definition at line 323 of file safe\_iterator.h.

#### 5.84.3.27 operator--() [2/2]

```
template<typename _Iterator, typename _Sequence>
__Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- (
    int ) [inline], [noexcept]
```

Iterator postdecrement.

##### Precondition

iterator is decrementable

Definition at line 338 of file safe\_iterator.h.

#### 5.84.3.28 operator->()

```
template<typename _Iterator, typename _Sequence>
pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> ( ) const [inline],
[noexcept]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 279 of file safe\_iterator.h.

#### 5.84.3.29 operator=() [1/2]

```
template<typename _Iterator, typename _Sequence>
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator= (
    const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline], [noexcept]
```

Copy assignment.

Definition at line 199 of file safe\_iterator.h.

#### 5.84.3.30 operator=() [2/2]

```
template<typename _Iterator, typename _Sequence>
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator= (
    _Safe_iterator< _Iterator, _Sequence > && __x ) [inline], [noexcept]
```

Move assignment.

##### Postcondition

\_\_x is singular and unattached

Definition at line 231 of file safe\_iterator.h.

#### 5.84.4 Member Data Documentation

5.84.4.1 `_M_next`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

5.84.4.2 `_M_prior`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

5.84.4.3 `_M_sequence`

```
_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_iterator_base::_M_attached_to()`, `__gnu_debug::Safe_sequence< map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`, `__gnu_debug::Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::Safe_local_iterator_base::_Safe_local_iterator_base()`.

5.84.4.4 `_M_version`

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

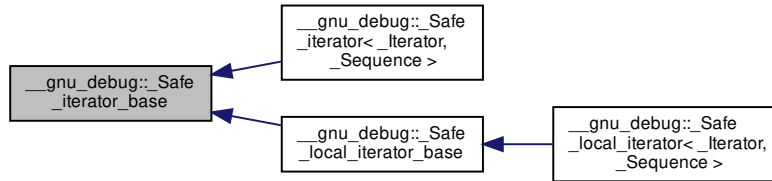
Referenced by `__gnu_debug::Safe_iterator_base::_M_invalidate()`, and `__gnu_debug::Safe_sequence< map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_iterator.h](#)
- [safe\\_iterator.tcc](#)

## 5.85 \_\_gnu\_debug::\_Safe\_iterator\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_iterator\_base:



### Public Member Functions

- [bool \\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \* \_\_seq) const
- [bool \\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) & \_\_x) const throw ()
- [void \\_M\\_detach\\_single](#) () throw ()
- [void \\_M\\_invalidate](#) ()
- [void \\_M\\_reset](#) () throw ()
- [bool \\_M\\_singular](#) () const throw ()
- [void \\_M\\_unlink](#) () throw ()

### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_next](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_prior](#)
- [\\_Safe\\_sequence\\_base](#) \* [\\_M\\_sequence](#)
- unsigned int [\\_M\\_version](#)

### Protected Member Functions

- [\\_Safe\\_iterator\\_base](#) ()
- [\\_Safe\\_iterator\\_base](#) (const [\\_Safe\\_sequence\\_base](#) \* \_\_seq, bool \_\_constant)
- [\\_Safe\\_iterator\\_base](#) (const [\\_Safe\\_iterator\\_base](#) & \_\_x, bool \_\_constant)
- [void \\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \* \_\_seq, bool \_\_constant)
- [void \\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \* \_\_seq, bool \_\_constant) throw ()
- [void \\_M\\_detach](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex & \\_M\\_get\\_mutex](#) () throw ()

### Friends

- class [\\_Safe\\_sequence\\_base](#)

### 5.85.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_base.h`.

### 5.85.2 Constructor & Destructor Documentation

#### 5.85.2.1 `_Safe_iterator_base()` [1/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( ) [inline], [protected]
```

Initializes the iterator and makes it singular.

Definition at line 78 of file `safe_base.h`.

#### 5.85.2.2 `_Safe_iterator_base()` [2/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
    const \_Safe\_sequence\_base * __seq,
    bool __constant ) [inline], [protected]
```

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 89 of file `safe_base.h`.

References `_M_attach()`.

#### 5.85.2.3 `_Safe_iterator_base()` [3/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
    const \_Safe\_iterator\_base & __x,
    bool __constant ) [inline], [protected]
```

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 96 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.



### 5.85.3 Member Function Documentation

#### 5.85.3.1 `_M_attach()`

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
    _Safe_sequence_base * __seq,
    bool __constant ) [protected]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_attach()`, and `_Safe_iterator_base()`.

#### 5.85.3.2 `_M_attach_single()`

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
    _Safe_sequence_base * __seq,
    bool __constant ) throw ( ) [protected]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_attach_single()`.

#### 5.85.3.3 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const _Safe_sequence_base * __seq ) const [inline]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `_M_sequence`.

#### 5.85.3.4 `_M_can_compare()`

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const _Safe_iterator_base & __x ) const throw ( )
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

#### 5.85.3.5 \_M\_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach ( ) [protected]
```

Detach the iterator for whatever sequence it is attached to, if any.

#### 5.85.3.6 \_M\_detach\_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single ( ) throw ( )
```

Likewise, but not thread-safe.

#### 5.85.3.7 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw ( ) [protected]
```

For use in \_Safe\_iterator.

#### 5.85.3.8 \_M\_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file safe\_base.h.

References \_M\_version.

#### 5.85.3.9 \_M\_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw ( )
```

Reset all member variables

#### 5.85.3.10 \_M\_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ( )
```

Is this iterator singular?

Referenced by \_\_gnu\_debug::\_check\_singular\_aux(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_incrementable(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_incrementable().

#### 5.85.3.11 `_M_unlink()`

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw ( )    [inline]
```

Unlink itself

Definition at line 155 of file `safe_base.h`.

### 5.85.4 Member Data Documentation

#### 5.85.4.1 `_M_next`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence` `!=` `NULL`.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.85.4.2 `_M_prior`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence` `!=` `NULL`.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.85.4.3 `_M_sequence`

```
_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence
```

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `_M_attached_to()`, `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `_Safe_iterator_base()`, and `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

5.85.4.4 `_M_version`

```
unsigned int __gnu_debug::Safe_iterator_base::_M_version
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

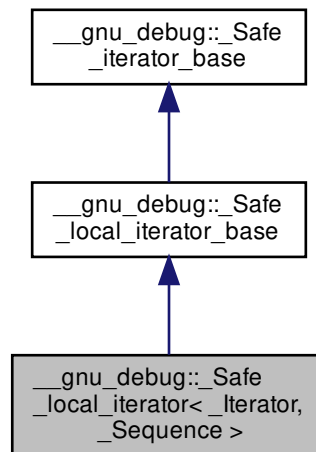
Referenced by `_M_invalidate()`, and `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

5.86 `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>`:



## Public Types

- `typedef _Traits::difference_type` **difference\_type**
- `typedef _Traits::iterator_category` **iterator\_category**
- `typedef _Iterator` **iterator\_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::value_type` **value\_type**

## Public Member Functions

- [\\_Safe\\_local\\_iterator](#) () noexcept
- [\\_Safe\\_local\\_iterator](#) (const [\\_Iterator](#) &\_\_i, const [\\_Safe\\_sequence\\_base](#) \*\_\_cont)
- [\\_Safe\\_local\\_iterator](#) (const [\\_Safe\\_local\\_iterator](#) &\_\_x) noexcept
- [\\_Safe\\_local\\_iterator](#) ([\\_Safe\\_local\\_iterator](#) &&\_\_x) noexcept
- template<typename [\\_MutableIterator](#) >  
[\\_Safe\\_local\\_iterator](#) (const [\\_Safe\\_local\\_iterator](#)< [\\_MutableIterator](#), typename [\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< std::\_\_↵  
[\\_are\\_same](#)< [\\_MutableIterator](#), typename [\\_Sequence::local\\_iterator::iterator\\_type](#) >::\_\_value, [\\_Sequence](#) >::\_\_↵  
[\\_type](#) > &\_\_x)
- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq)
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- bool [\\_M\\_dereferenceable](#) () const
- [\\_\\_gnu\\_cxx::\\_\\_conditional\\_type](#)< std::\_\_are\_same< [\\_Const\\_local\\_iterator](#), [\\_Safe\\_local\\_iterator](#) >::\_\_value,  
const [\\_Sequence](#) \*, [\\_Sequence](#) \* >::\_\_type [\\_M\\_get\\_sequence](#) () const
- template<typename [\\_Other](#) >  
bool [\\_M\\_in\\_same\\_bucket](#) (const [\\_Safe\\_local\\_iterator](#)< [\\_Other](#), [\\_Sequence](#) > &\_\_other) const
- bool [\\_M\\_incrementable](#) () const
- void [\\_M\\_invalidate](#) ()
- bool [\\_M\\_is\\_begin](#) () const
- bool [\\_M\\_is\\_end](#) () const
- void [\\_M\\_reset](#) () throw ()
- bool [\\_M\\_singular](#) () const throw ()
- void [\\_M\\_unlink](#) () throw ()
- bool [\\_M\\_valid\\_range](#) (const [\\_Safe\\_local\\_iterator](#) &\_\_rhs, [std::pair](#)< [difference\\_type](#), [\\_Distance\\_precision](#) >  
&\_\_dist\_info) const
- [\\_Iterator](#) & [base](#) () noexcept
- const [\\_Iterator](#) & [base](#) () const noexcept
- [size\\_type](#) [bucket](#) () const
- [operator \\_Iterator](#) () const
- reference [operator\\*](#) () const
- [\\_Safe\\_local\\_iterator](#) & [operator++](#) ()
- [\\_Safe\\_local\\_iterator](#) [operator++](#) (int)
- pointer [operator->](#) () const
- [\\_Safe\\_local\\_iterator](#) & [operator=](#) (const [\\_Safe\\_local\\_iterator](#) &\_\_x)
- [\\_Safe\\_local\\_iterator](#) & [operator=](#) ([\\_Safe\\_local\\_iterator](#) &&\_\_x) noexcept

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_next](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_prior](#)
- [\\_Safe\\_sequence\\_base](#) \* [\\_M\\_sequence](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void `_M_attach` (`_Safe_sequence_base` \*`__seq`, bool `__constant`)
- void `_M_attach_single` (`_Safe_sequence_base` \*`__seq`, bool `__constant`) throw ()
- void `_M_detach` ()
- void `_M_detach_single` () throw ()
- `_Safe_unordered_container_base` \* `_M_get_container` () const noexcept
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()

## 5.86.1 Detailed Description

```
template<typename _Iterator, typename _Sequence>
class __gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>
```

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 59 of file `formatter.h`.

## 5.86.2 Constructor &amp; Destructor Documentation

5.86.2.1 `_Safe_local_iterator()` [1/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>::__Safe_local_iterator ( ) [inline],
[noexcept]
```

## Postcondition

the iterator is singular and unattached

Definition at line 84 of file `safe_local_iterator.h`.

**5.86.2.2** `_Safe_local_iterator()` [2/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
    const _Iterator & __i,
    const _Safe_sequence_base * __cont ) [inline]
```

Safe iterator construction from an unsafe iterator and its sequence.

**Precondition**

`seq` is not NULL

**Postcondition**

this is not singular

Definition at line 93 of file `safe_local_iterator.h`.

**5.86.2.3** `_Safe_local_iterator()` [3/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
    const _Safe_local_iterator< _Iterator, _Sequence > & __x ) [inline], [noexcept]
```

Copy construction.

Definition at line 105 of file `safe_local_iterator.h`.

**5.86.2.4** `_Safe_local_iterator()` [4/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
    _Safe_local_iterator< _Iterator, _Sequence > && __x ) [inline], [noexcept]
```

Move construction.

**Postcondition**

`__x` is singular and unattached

Definition at line 122 of file `safe_local_iterator.h`.

## 5.86.2.5 \_Safe\_local\_iterator() [5/5]

```
template<typename _Iterator , typename _Sequence >
template<typename _MutableIterator >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
    const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< std::
::__are_same< _MutableIterator, typename _Sequence::local_iterator::iterator_type >::__value, _
Sequence >::__type > & __x ) [inline]
```

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 141 of file safe\_local\_iterator.h.

## 5.86.3 Member Function Documentation

## 5.86.3.1 \_M\_attach() [1/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
    _Safe_sequence_base * __seq,
    bool __constant ) [protected], [inherited]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_attach(), and \_\_gnu\_debug::\_Safe\_local\_iterator\_base::\_Safe\_local\_iterator\_base().

## 5.86.3.2 \_M\_attach() [2/2]

```
template<typename _Iterator , typename _Sequence >
void __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach (
    _Safe_sequence_base * __seq ) [inline]
```

Attach iterator to the given sequence.

Definition at line 304 of file safe\_local\_iterator.h.

References \_\_gnu\_debug::\_Safe\_local\_iterator\_base::\_M\_attach().



#### 5.86.3.3 `_M_attach_single()` [1/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
    _Safe_sequence_base * __seq,
    bool __constant ) throw ( )    [protected], [inherited]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence >::_M_attach_single()`.

#### 5.86.3.4 `_M_attach_single()` [2/2]

```
template<typename _Iterator , typename _Sequence >
void __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence >::_M_attach_single (
    _Safe_sequence_base * __seq )    [inline]
```

Likewise, but not thread-safe.

Definition at line 309 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator_base::_M_attach_single()`.

#### 5.86.3.5 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const _Safe_sequence_base * __seq ) const    [inline], [inherited]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

#### 5.86.3.6 `_M_can_compare()`

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const _Safe_iterator_base & __x ) const throw ( )    [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

### 5.86.3.7 \_M\_dereferenceable()

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable ( ) const
[inline]
```

Is the iterator dereferenceable?

Definition at line 314 of file safe\_local\_iterator.h.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

### 5.86.3.8 \_M\_detach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach ( ) [protected], [inherited]
```

Detach the iterator for whatever container it is attached to, if any.

### 5.86.3.9 \_M\_detach\_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ( ) throw ( ) [protected], [inherited]
```

Likewise, but not thread-safe.

### 5.86.3.10 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in `_Safe_iterator`.

### 5.86.3.11 \_M\_in\_same\_bucket()

```
template<typename _Iterator , typename _Sequence >
template<typename _Other >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_in_same_bucket (
    const _Safe_local_iterator< _Other, _Sequence > & __other ) const [inline]
```

Is this iterator part of the same bucket as the other one?

Definition at line 348 of file safe\_local\_iterator.h.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket()`.

#### 5.86.3.12 `_M_incrementable()`

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable ( ) const [inline]
```

Is the iterator incrementable?

Definition at line 319 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

#### 5.86.3.13 `_M_invalidate()`

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_version`.

#### 5.86.3.14 `_M_is_begin()`

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_begin ( ) const [inline]
```

Is this iterator equal to the sequence's `begin(bucket)` iterator?

Definition at line 338 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket()`.

#### 5.86.3.15 `_M_is_end()`

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end ( ) const [inline]
```

Is this iterator equal to the sequence's `end(bucket)` iterator?

Definition at line 342 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket()`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable()`.

**5.86.3.16** `_M_reset()`

```
void __gnu_debug::Safe_iterator_base::_M_reset ( ) throw ( ) [inherited]
```

Reset all member variables

**5.86.3.17** `_M_singular()`

```
bool __gnu_debug::Safe_iterator_base::_M_singular ( ) const throw ( ) [inherited]
```

Is this iterator singular?

Referenced by `__gnu_debug::check_singular_aux()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

**5.86.3.18** `_M_unlink()`

```
void __gnu_debug::Safe_iterator_base::_M_unlink ( ) throw ( ) [inline], [inherited]
```

Unlink itself

Definition at line 155 of file `safe_base.h`.

**5.86.3.19** `base()`

```
template<typename _Iterator , typename _Sequence >
__Iterator& __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::base ( ) [inline], [noexcept]
```

Return the underlying iterator.

Definition at line 285 of file `safe_local_iterator.h`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_begin()`, `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::bucket()`.

#### 5.86.3.20 bucket()

```
template<typename _Iterator , typename _Sequence >
size_type __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket ( ) const [inline]
```

Return the bucket.

Definition at line 294 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base()`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_in_same_bucket()`, `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_is_begin()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_is_end()`.

#### 5.86.3.21 operator\_iterator()

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator_iterator ( ) const [inline]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 300 of file `safe_local_iterator.h`.

#### 5.86.3.22 operator\*()

```
template<typename _Iterator , typename _Sequence >
reference __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator* ( ) const [inline]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 228 of file `safe_local_iterator.h`.

#### 5.86.3.23 operator++() [1/2]

```
template<typename _Iterator , typename _Sequence >
_Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ ( )
[inline]
```

Iterator preincrement.

##### Precondition

iterator is incrementable

Definition at line 255 of file `safe_local_iterator.h`.

#### 5.86.3.24 `operator++()` [2/2]

```
template<typename _Iterator , typename _Sequence >
_Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ (
    int ) [inline]
```

Iterator postincrement.

##### Precondition

iterator is incrementable

Definition at line 270 of file `safe_local_iterator.h`.

#### 5.86.3.25 `operator->()`

```
template<typename _Iterator , typename _Sequence >
pointer __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator-> ( ) const [inline]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 241 of file `safe_local_iterator.h`.

#### 5.86.3.26 `operator=()` [1/2]

```
template<typename _Iterator , typename _Sequence >
_Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator= (
    const _Safe_local_iterator< _Iterator, _Sequence > & __x ) [inline]
```

Copy assignment.

Definition at line 163 of file `safe_local_iterator.h`.

#### 5.86.3.27 `operator=()` [2/2]

```
template<typename _Iterator , typename _Sequence >
_Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator= (
    _Safe_local_iterator< _Iterator, _Sequence > && __x ) [inline], [noexcept]
```

Move assignment.

##### Postcondition

`__x` is singular and unattached

Definition at line 194 of file `safe_local_iterator.h`.

## 5.86.4 Member Data Documentation

### 5.86.4.1 `_M_next`

`_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next` [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.86.4.2 `_M_prior`

`_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.86.4.3 `_M_sequence`

`_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

### 5.86.4.4 `_M_version`

`unsigned int __gnu_debug::_Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

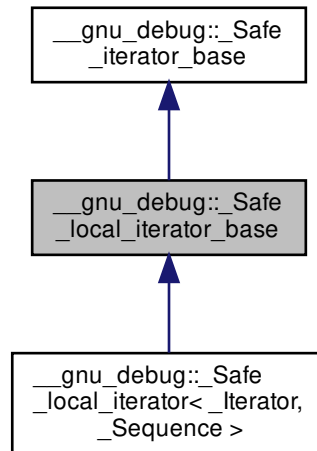
Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`, and `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_local\\_iterator.h](#)
- [safe\\_local\\_iterator.tcc](#)

## 5.87 \_\_gnu\_debug::\_Safe\_local\_iterator\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_local\_iterator\_base:



## Public Member Functions

- [bool](#) [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \* \_\_seq) const
- [bool](#) [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) & \_\_x) const throw ()
- [void](#) [\\_M\\_invalidate](#) ()
- [void](#) [\\_M\\_reset](#) () throw ()
- [bool](#) [\\_M\\_singular](#) () const throw ()
- [void](#) [\\_M\\_unlink](#) () throw ()

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_next](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_prior](#)
- [\\_Safe\\_sequence\\_base](#) \* [\\_M\\_sequence](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- [\\_Safe\\_local\\_iterator\\_base](#) ()
- [\\_Safe\\_local\\_iterator\\_base](#) (const [\\_Safe\\_sequence\\_base](#) \* \_\_seq, bool \_\_constant)
- [\\_Safe\\_local\\_iterator\\_base](#) (const [\\_Safe\\_local\\_iterator\\_base](#) & \_\_x, bool \_\_constant)
- [void](#) [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \* \_\_seq, bool \_\_constant)
- [void](#) [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \* \_\_seq, bool \_\_constant) throw ()
- [void](#) [\\_M\\_detach](#) ()
- [void](#) [\\_M\\_detach\\_single](#) () throw ()
- [\\_Safe\\_unordered\\_container\\_base](#) \* [\\_M\\_get\\_container](#) () const noexcept
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()



### 5.87.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_unordered_base.h`.

### 5.87.2 Constructor & Destructor Documentation

#### 5.87.2.1 `_Safe_local_iterator_base()` [1/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ( ) [inline], [protected]
```

Initializes the iterator and makes it singular.

Definition at line 54 of file `safe_unordered_base.h`.

#### 5.87.2.2 `_Safe_local_iterator_base()` [2/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
    const _Safe_sequence_base * __seq,
    bool __constant ) [inline], [protected]
```

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 64 of file `safe_unordered_base.h`.

References `_M_attach()`.

#### 5.87.2.3 `_Safe_local_iterator_base()` [3/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
    const _Safe_local_iterator_base & __x,
    bool __constant ) [inline], [protected]
```

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 70 of file `safe_unordered_base.h`.

References `_M_attach()`, and `__gnu_debug::_Safe_iterator_base::_M_sequence`.

## 5.87.3 Member Function Documentation

## 5.87.3.1 \_M\_attach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
    _Safe_sequence_base * __seq,
    bool __constant ) [protected]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_attach()`, and `_Safe_local_iterator_base()`.

## 5.87.3.2 \_M\_attach\_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
    _Safe_sequence_base * __seq,
    bool __constant ) throw ( ) [protected]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_attach_single()`.

## 5.87.3.3 \_M\_attached\_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
    const _Safe_sequence_base * __seq ) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

## 5.87.3.4 \_M\_can\_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
    const _Safe_iterator_base & __x ) const throw ( ) [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

#### 5.87.3.5 `_M_detach()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach ( ) [protected]
```

Detach the iterator for whatever container it is attached to, if any.

#### 5.87.3.6 `_M_detach_single()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ( ) throw ( ) [protected]
```

Likewise, but not thread-safe.

#### 5.87.3.7 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in `_Safe_iterator`.

#### 5.87.3.8 `_M_invalidate()`

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_version`.

#### 5.87.3.9 `_M_reset()`

```
void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw ( ) [inherited]
```

Reset all member variables

#### 5.87.3.10 `_M_singular()`

```
bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ( ) [inherited]
```

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

### 5.87.3.11 \_M\_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw ( )    [inline], [inherited]
```

Unlink itself

Definition at line 155 of file `safe_base.h`.

## 5.87.4 Member Data Documentation

### 5.87.4.1 \_M\_next

```
\_Safe\_iterator\_base\* __gnu_debug::_Safe_iterator_base::_M_next    [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.87.4.2 \_M\_prior

```
\_Safe\_iterator\_base\* __gnu_debug::_Safe_iterator_base::_M_prior    [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.87.4.3 \_M\_sequence

```
\_Safe\_sequence\_base\* __gnu_debug::_Safe_iterator_base::_M_sequence    [inherited]
```

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `_Safe_local_iterator_base()`.

#### 5.87.4.4 `_M_version`

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

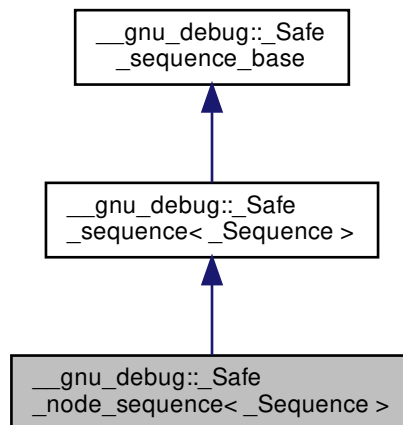
Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`, and `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

#### 5.88 `__gnu_debug::_Safe_node_sequence< _Sequence >` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_node_sequence< _Sequence >`:



#### Public Member Functions

- `template<typename _Predicate >`  
`void \_M\_invalidate\_if (_Predicate __pred)`
- `template<typename _Predicate >`  
`void \_M\_transfer\_from\_if (\_Safe\_sequence &__from, _Predicate __pred)`

**Public Attributes**

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

**Protected Member Functions**

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x) noexcept`

**5.88.1 Detailed Description**

```
template<typename _Sequence>
class __gnu_debug::_Safe_node_sequence<_Sequence>
```

Like `_Safe_sequence` but with a special `_M_invalidate_all` implementation not invalidating past-the-end iterators. Used by node based sequence.

Definition at line 131 of file `safe_sequence.h`.

**5.88.2 Member Function Documentation****5.88.2.1 `_M_detach_all()`**

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

**5.88.2.2 `_M_detach_singular()`**

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

**Postcondition**

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

### 5.88.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( )    [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.88.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const    [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

### 5.88.2.5 `_M_invalidate_if()`

```
template<typename _Sequence >  
template<typename _Predicate >  
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (   
    _Predicate __pred )    [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

### 5.88.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )    [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### 5.88.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (   
    _Safe_sequence_base & __x )    [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.88.2.8 `_M_transfer_from_if()`

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if (
    Safe_sequence<_Sequence> & __from,
    _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

## 5.88.3 Member Data Documentation

### 5.88.3.1 `_M_const_iterators`

```
Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

### 5.88.3.2 `_M_iterators`

```
Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

### 5.88.3.3 `_M_version`

```
unsigned int __gnu_debug::Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence_base::_M_invalidate_all()`.

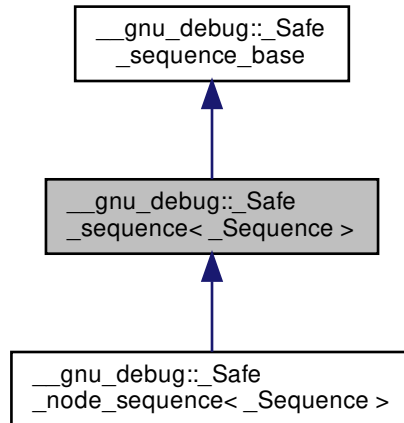
The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)



## 5.89 \_\_gnu\_debug::\_Safe\_sequence<\_Sequence> Class Template Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_sequence<\_Sequence>:



### Public Member Functions

- `template<typename _Predicate>`  
`void \_M\_invalidate\_if (_Predicate __pred)`
- `template<typename _Predicate>`  
`void \_M\_transfer\_from\_if (\_Safe\_sequence &__from, _Predicate __pred)`

### Public Attributes

- `\_Safe\_iterator\_base * \_M\_const\_iterators`
- `\_Safe\_iterator\_base * \_M\_iterators`
- `unsigned int \_M\_version`

### Protected Member Functions

- `void \_M\_detach\_all ()`
- `void \_M\_detach\_singular ()`
- `\_\_gnu\_cxx::\_\_mutex & \_M\_get\_mutex () throw ()`
- `void \_M\_invalidate\_all () const`
- `void \_M\_revalidate\_singular ()`
- `void \_M\_swap (\_Safe\_sequence\_base &__x) noexcept`

## 5.89.1 Detailed Description

```
template<typename _Sequence>
class __gnu_debug::_Safe_sequence<_Sequence>
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 62 of file `formatter.h`.

## 5.89.2 Member Function Documentation

5.89.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

5.89.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

**Postcondition**

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.89.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

#### 5.89.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 5.89.2.5 `_M_invalidate_if()`

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
    _Predicate __pred )
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 5.89.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.89.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 5.89.2.8 `_M_transfer_from_if()`

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
    _Safe_sequence< _Sequence > & __from,
    _Predicate __pred )
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 5.89.3 Member Data Documentation

#### 5.89.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

#### 5.89.3.2 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

#### 5.89.3.3 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

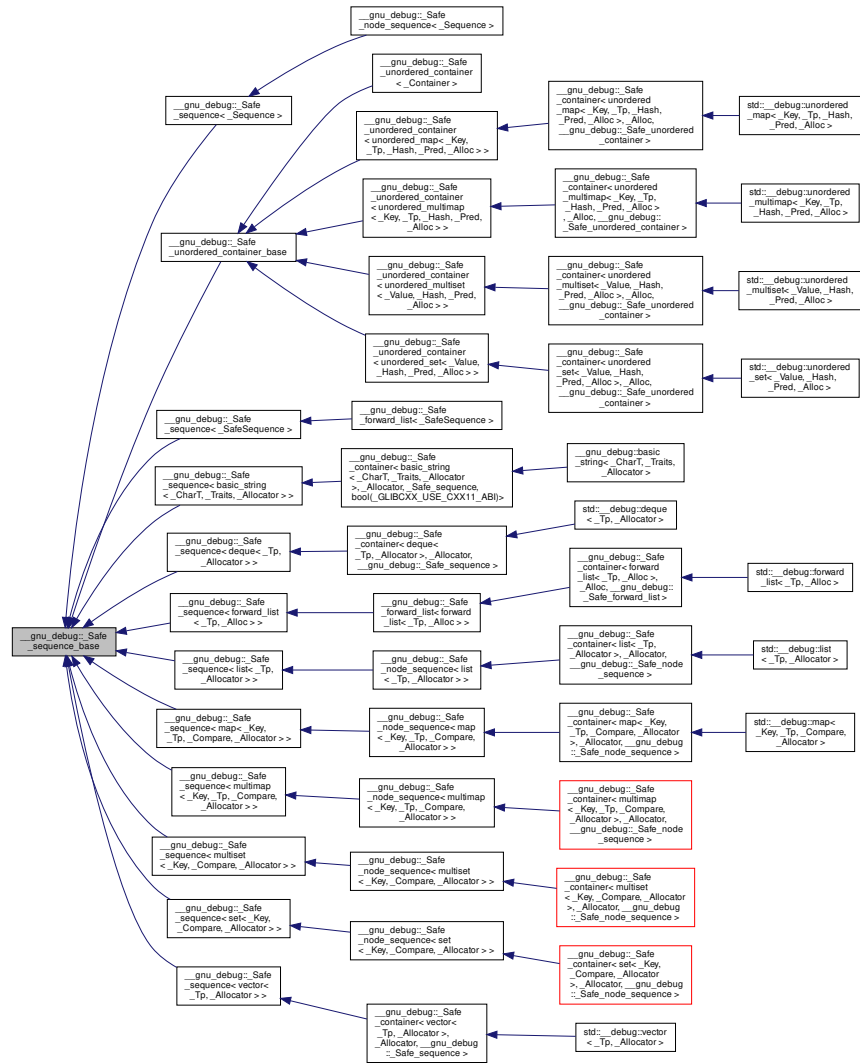
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_sequence.h](#)
- [safe\\_sequence.tcc](#)

## 5.90 \_\_gnu\_debug::Safe\_sequence\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::Safe\_sequence\_base:



### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

### Protected Member Functions

- [\\_Safe\\_sequence\\_base](#) (const [\\_Safe\\_sequence\\_base](#) &) noexcept

- `_Safe_sequence_base` (`_Safe_sequence_base` &&\_\_seq) noexcept
- `~_Safe_sequence_base` ()
- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `void _M_invalidate_all` () const
- `void _M_revalidate_singular` ()
- `void _M_swap` (`_Safe_sequence_base` &\_\_x) noexcept

#### Friends

- `class _Safe_iterator_base`

#### 5.90.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 188 of file `safe_base.h`.

#### 5.90.2 Constructor & Destructor Documentation

##### 5.90.2.1 `~_Safe_sequence_base()`

```
__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base ( ) [inline], [protected]
```

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 220 of file `safe_base.h`.

References `_M_detach_all()`.

#### 5.90.3 Member Function Documentation

#### 5.90.3.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected]
```

Detach all iterators, leaving them singular.

Referenced by `~_Safe_sequence_base()`.

#### 5.90.3.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 5.90.3.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.90.3.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `_M_version`.

#### 5.90.3.5 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### 5.90.3.6 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.90.4 Member Data Documentation

### 5.90.4.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.90.4.2 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.90.4.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `_M_invalidate_all()`.

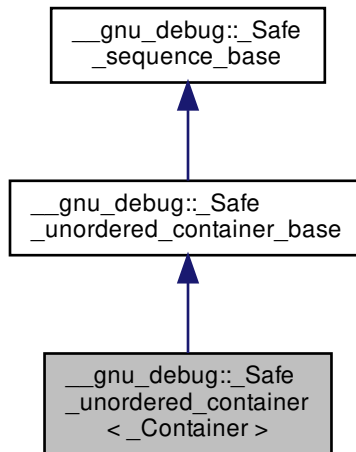
The documentation for this class was generated from the following file:

- [safe\\_base.h](#)



### 5.91 `__gnu_debug::_Safe_unordered_container<_Container>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_unordered_container<_Container>`:



#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void _M_invalidate_all () const`
- `template<typename _Predicate>  
void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate>  
void _M_invalidate_local_if (_Predicate __pred)`
- `void _M_invalidate_locals ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_unordered_container_base &__x) noexcept`
- `void _M_swap (_Safe_sequence_base &__x) noexcept`

### 5.91.1 Detailed Description

```
template<typename _Container>
class __gnu_debug::_Safe_unordered_container<_Container>
```

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

Definition at line 58 of file `safe_unordered_container.h`.

### 5.91.2 Member Function Documentation

#### 5.91.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

#### 5.91.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 5.91.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

#### 5.91.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 5.91.2.5 `_M_invalidate_if()`

```
template<typename _Container >
template<typename _Predicate >
void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_if (
    _Predicate __pred ) [protected]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

#### 5.91.2.6 `_M_invalidate_local_if()`

```
template<typename _Container >
template<typename _Predicate >
void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if (
    _Predicate __pred ) [protected]
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

#### 5.91.2.7 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.91.2.8 `_M_swap()` [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    _Safe_unordered_container_base & __x ) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.91.2.9 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.91.3 Member Data Documentation

### 5.91.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

### 5.91.3.2 \_M\_const\_local\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file safe\_unordered\_base.h.

### 5.91.3.3 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

### 5.91.3.4 `_M_local_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

### 5.91.3.5 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

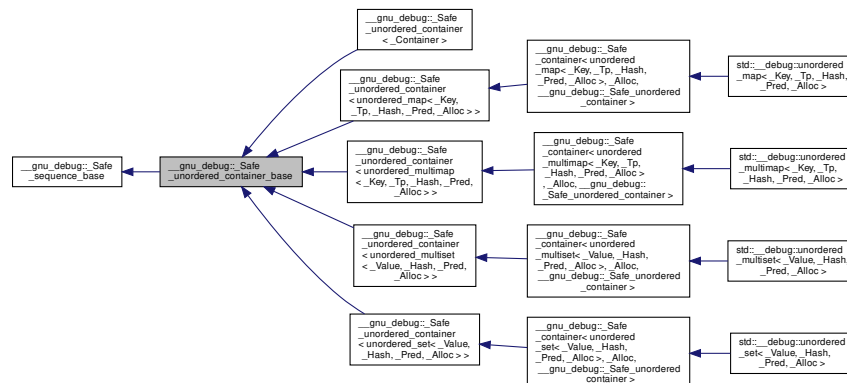
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following files:

- [safe\\_unordered\\_container.h](#)
- [safe\\_unordered\\_container.tcc](#)

## 5.92 `__gnu_debug::_Safe_unordered_container_base` Class Reference

Inheritance diagram for `__gnu_debug::_Safe_unordered_container_base`:



### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `_Safe_unordered_container_base` (const `_Safe_unordered_container_base` &) noexcept
- `_Safe_unordered_container_base` (`_Safe_unordered_container_base` &&\_\_x) noexcept
- `~_Safe_unordered_container_base` () noexcept
- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_invalidate_all` () const
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_unordered_container_base` &\_\_x) noexcept
- void `_M_swap` (`_Safe_sequence_base` &\_\_x) noexcept

## Friends

- class `_Safe_local_iterator_base`

## 5.92.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 120 of file `safe_unordered_base.h`.

## 5.92.2 Constructor &amp; Destructor Documentation

5.92.2.1 `~_Safe_unordered_container_base()`

```
__gnu_debug::_Safe_unordered_container_base::~~_Safe_unordered_container_base ( ) [inline], [protected],
[noexcept]
```

Notify all iterators that reference this container that the container is being destroyed.

Definition at line 151 of file `safe_unordered_base.h`.

## 5.92.3 Member Function Documentation

#### 5.92.3.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected]
```

Detach all iterators, leaving them singular.

#### 5.92.3.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

#### 5.92.3.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.92.3.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 5.92.3.5 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.92.3.6 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    _Safe_unordered_container_base & __x ) [protected], [noexcept]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.92.3.7 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.92.4 Member Data Documentation

## 5.92.4.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.92.4.2 \_M\_const\_local\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators
```

The list of constant local iterators that reference this container.

Definition at line 130 of file safe\_unordered\_base.h.

## 5.92.4.3 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().



#### 5.92.4.4 `_M_local_iterators`

```
\_Safe\_iterator\_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

#### 5.92.4.5 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

### 5.93 `__gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence >` Class Template Reference

#### Protected Member Functions

- `_Safe_vector` (const [\\_Safe\\_vector](#) &) noexcept
- `_Safe_vector` (size\_type \_\_n) noexcept
- `_Safe_vector` ([\\_Safe\\_vector](#) &&\_\_x) noexcept
- bool `_M_requires_reallocation` (size\_type \_\_elements) const noexcept
- void `_M_update_guaranteed_capacity` () noexcept
- [\\_Safe\\_vector](#) & `operator=` (const [\\_Safe\\_vector](#) &) noexcept
- [\\_Safe\\_vector](#) & `operator=` ([\\_Safe\\_vector](#) &&\_\_x) noexcept

#### Protected Attributes

- size\_type `_M_guaranteed_capacity`

#### 5.93.1 Detailed Description

```
template<typename _SafeSequence, typename _BaseSequence>
class __gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence >
```

Base class for Debug Mode vector.

Adds information about the guaranteed capacity, which is useful for detecting code which relies on non-portable implementation details of the libstdc++ reallocation policy.

Definition at line 50 of file `debug/vector`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

## 5.94 \_\_gnu\_debug::Sequence\_traits&lt; \_Sequence &gt; Struct Template Reference

## Public Types

- typedef \_Distance\_traits< typename \_Sequence::iterator > **\_DistTraits**

## Static Public Member Functions

- static [\\_DistTraits::\\_type](#) **\_S\_size** (const \_Sequence &\_\_seq)

## 5.94.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::Sequence_traits< _Sequence >
```

Sequence traits giving the size of a container if possible.

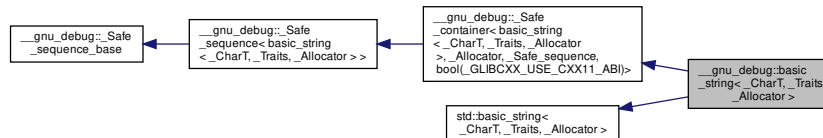
Definition at line 60 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 5.95 \_\_gnu\_debug::basic\_string&lt; \_CharT, \_Traits, \_Allocator &gt; Class Template Reference

Inheritance diagram for `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`:



## Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#)< typename \_Base::const\_iterator, [basic\\_string](#) > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< [const\\_iterator](#) > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#)< typename \_Base::iterator, [basic\\_string](#) > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< [iterator](#) > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Traits **traits\_type**
- typedef \_Traits::char\_type **value\_type**

## Public Member Functions

- **basic\_string** (const \_Allocator &\_\_a) noexcept
- **basic\_string** (const basic\_string &)=default
- **basic\_string** (basic\_string &&)=default
- **basic\_string** (std::initializer\_list<\_CharT> \_\_l, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (\_Base &&\_\_base) noexcept
- **basic\_string** (const \_Base &\_\_base)
- **basic\_string** (const basic\_string &\_\_str, size\_type \_\_pos, size\_type \_\_n=\_Base::npos, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (const \_CharT \* \_\_s, size\_type \_\_n, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (const \_CharT \* \_\_s, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (size\_type \_\_n, \_CharT \_\_c, const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator>
 **basic\_string** (\_InputIterator \_\_begin, \_InputIterator \_\_end, const \_Allocator &\_\_a=\_Allocator())
- **\_Base & \_M\_base** () noexcept
- const **\_Base & \_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- **basic\_string**<\_CharT, \_Traits, \_Allocator> & **\_M\_replace\_dispatch** (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2, \_\_false\_type)
- void **\_M\_swap** (\_Safe\_container &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- \_CharT \* **\_S\_construct** (\_InIterator \_\_beg, \_InIterator \_\_end, const \_Allocator &\_\_a, forward\_iterator\_tag)
- **basic\_string & append** (const basic\_string &\_\_str)
- **basic\_string & append** (const basic\_string &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- **basic\_string & append** (const \_CharT \* \_\_s, size\_type \_\_n)
- **basic\_string & append** (const \_CharT \* \_\_s)
- **basic\_string & append** (size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator>
 **basic\_string & append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **basic\_string & append** (const basic\_string &\_\_str)
- **basic\_string & append** (const basic\_string &\_\_str, size\_type \_\_pos, size\_type \_\_n=npos)
- **basic\_string & append** (initializer\_list<\_CharT> \_\_l)
- **basic\_string & assign** (const basic\_string &\_\_x)
- **basic\_string & assign** (basic\_string &&\_\_x) noexcept(noexcept(std::declval<\_Base &>().assign(std::move(\_\_x))))
- **basic\_string & assign** (const basic\_string &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- **basic\_string & assign** (const \_CharT \* \_\_s, size\_type \_\_n)
- **basic\_string & assign** (const \_CharT \* \_\_s)
- **basic\_string & assign** (size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator>
 **basic\_string & assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **basic\_string & assign** (std::initializer\_list<\_CharT> \_\_l)
- **basic\_string & assign** (const basic\_string &\_\_str)
- **basic\_string & assign** (basic\_string &&\_\_str)
- **basic\_string & assign** (const basic\_string &\_\_str, size\_type \_\_pos, size\_type \_\_n=npos)
- const\_reference **at** (size\_type \_\_n) const
- reference **at** (size\_type \_\_n)
- reference **back** ()
- const\_reference **back** () const noexcept
- **iterator begin** ()

- `const_iterator begin ()` const noexcept
- `const _CharT * c_str ()` const noexcept
- `size_type capacity ()` const noexcept
- `const_iterator cbegin ()` const noexcept
- `const_iterator cend ()` const noexcept
- `void clear ()`
- `int compare (const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2) const`
- `int compare (const _CharT * __s) const`
- `int compare (size_type __pos1, size_type __n1, const _CharT * __s) const`
- `int compare (size_type __pos1, size_type __n1, const _CharT * __s, size_type __n2) const`
- `int compare (const basic_string &__str) const`
- `int compare (size_type __pos, size_type __n, const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const`
- `size_type copy (_CharT * __s, size_type __n, size_type __pos=0) const`
- `const_reverse_iterator crbegin ()` const noexcept
- `const_reverse_iterator crend ()` const noexcept
- `const _CharT * data ()` const noexcept
- `bool empty ()` const noexcept
- `iterator end ()`
- `const_iterator end ()` const noexcept
- `basic_string & erase (size_type __pos=0, size_type __n=_Base::npos)`
- `iterator erase (iterator __position)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `iterator erase (iterator __first, iterator __last)`
- `size_type find (const basic_string &__str, size_type __pos=0) const` noexcept
- `size_type find (const _CharT * __s, size_type __pos, size_type __n) const`
- `size_type find (const _CharT * __s, size_type __pos=0) const`
- `size_type find (_CharT __c, size_type __pos=0) const` noexcept
- `size_type find (const basic_string &__str, size_type __pos=0) const` noexcept
- `size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const` noexcept
- `size_type find_first_not_of (const _CharT * __s, size_type __pos, size_type __n) const`
- `size_type find_first_not_of (const _CharT * __s, size_type __pos=0) const`
- `size_type find_first_not_of (_CharT __c, size_type __pos=0) const` noexcept
- `size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const` noexcept
- `size_type find_first_of (const basic_string &__str, size_type __pos=0) const` noexcept
- `size_type find_first_of (const _CharT * __s, size_type __pos, size_type __n) const`
- `size_type find_first_of (const _CharT * __s, size_type __pos=0) const`
- `size_type find_first_of (_CharT __c, size_type __pos=0) const` noexcept
- `size_type find_first_of (const basic_string &__str, size_type __pos=0) const` noexcept
- `size_type find_last_not_of (const basic_string &__str, size_type __pos=_Base::npos) const` noexcept
- `size_type find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const`
- `size_type find_last_not_of (const _CharT * __s, size_type __pos=_Base::npos) const`
- `size_type find_last_not_of (_CharT __c, size_type __pos=_Base::npos) const` noexcept
- `size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const` noexcept
- `size_type find_last_of (const basic_string &__str, size_type __pos=_Base::npos) const` noexcept
- `size_type find_last_of (const _CharT * __s, size_type __pos, size_type __n) const`

- size\_type **find\_last\_of** (const \_CharT \*\_\_s, size\_type \_\_pos=[\\_Base::npos](#)) const
- size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=[\\_Base::npos](#)) const noexcept
- size\_type **find\_last\_of** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- reference **front** ()
- const\_reference **front** () const noexcept
- allocator\_type **get\_allocator** () const noexcept
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, const \_CharT \*\_\_s)
- [basic\\_string](#) & **insert** (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- iterator **insert** (iterator \_\_p, \_CharT \_\_c)
- void **insert** (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator >  
void **insert** (iterator \_\_p, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (iterator \_\_p, [std::initializer\\_list](#)< \_CharT > \_\_l)
- void **insert** (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **insert** (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void **insert** (iterator \_\_p, [initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **insert** (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n=[npos](#))
- iterator **insert** (iterator \_\_p, \_CharT \_\_c)
- size\_type **length** () const noexcept
- size\_type **max\_size** () const noexcept
- [basic\\_string](#) & **operator+=** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **operator+=** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **operator+=** (\_CharT \_\_c)
- [basic\\_string](#) & **operator+=** ([std::initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & **operator+=** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **operator=** (const [basic\\_string](#) &)=default
- [basic\\_string](#) & **operator=** ([basic\\_string](#) &&)=default
- [basic\\_string](#) & **operator=** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **operator=** (\_CharT \_\_c)
- [basic\\_string](#) & **operator=** ([std::initializer\\_list](#)< \_CharT > \_\_l)
- const\_reference **operator[]** (size\_type \_\_pos) const noexcept
- reference **operator[]** (size\_type \_\_pos)
- void **pop\_back** ()
- void **push\_back** (\_CharT \_\_c)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const noexcept
- [basic\\_string](#) & **replace** (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **replace** (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)

- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `size_type` \_\_n, `_CharT` \_\_c)
- `template<typename _InputIterator>`  
`basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `_InputIterator` \_\_j1, `_InputIterator` \_\_j2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `std::initializer_list<_CharT>` \_\_l)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n, `const basic_string` & \_\_str)
- `basic_string` & `replace` (`size_type` \_\_pos1, `size_type` \_\_n1, `const basic_string` & \_\_str, `size_type` \_\_pos2, `size_type` \_\_n2=`npos`)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `const basic_string` & \_\_str)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `const _CharT *` \_\_s, `size_type` \_\_n)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `const _CharT *` \_\_s)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `size_type` \_\_n, `_CharT` \_\_c)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `_InputIterator` \_\_k1, `_InputIterator` \_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `_CharT *` \_\_k1, `_CharT *` \_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `const _CharT *` \_\_k1, `const _CharT *` \_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `iterator` \_\_k1, `iterator` \_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `const_iterator` \_\_k1, `const_iterator` \_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `initializer_list<_CharT>` \_\_l)
- `void reserve` (`size_type` \_\_res\_arg=0)
- `void resize` (`size_type` \_\_n, `_CharT` \_\_c)
- `void resize` (`size_type` \_\_n)
- `size_type rfind` (`const basic_string` & \_\_str, `size_type` \_\_pos=`_Base::npos`) `const noexcept`
- `size_type rfind` (`const _CharT *` \_\_s, `size_type` \_\_pos, `size_type` \_\_n) `const`
- `size_type rfind` (`const _CharT *` \_\_s, `size_type` \_\_pos=`_Base::npos`) `const`
- `size_type rfind` (`_CharT` \_\_c, `size_type` \_\_pos=`_Base::npos`) `const noexcept`
- `size_type rfind` (`const basic_string` & \_\_str, `size_type` \_\_pos=`npos`) `const noexcept`
- `void shrink_to_fit` () `noexcept`
- `size_type size` () `const noexcept`
- `basic_string substr` (`size_type` \_\_pos=0, `size_type` \_\_n=`_Base::npos`) `const`
- `void swap` (`basic_string` & \_\_x) `noexcept` (/\*conditional \*/)
- `void swap` (`basic_string` & \_\_s)

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- `unsigned int` `_M_version`

#### Static Public Attributes

- `static const size_type` `npos`

#### Protected Member Functions

- `void` `_M_detach_all` ()
- `void` `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () `throw` ()
- `void` `_M_invalidate_all` () `const`
- `void` `_M_revalidate_singular` ()
- `_Safe_container` & `_M_safe` () `noexcept`
- `void` `_M_swap` (`_Safe_sequence_base` & \_\_x) `noexcept`

### 5.95.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
class __gnu_debug::basic_string<_CharT, _Traits, _Allocator>
```

Class std::basic\_string with safety/checking/debug instrumentation.

Definition at line 44 of file debug/string.

### 5.95.2 Member Function Documentation

#### 5.95.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

#### 5.95.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

#### 5.95.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator> >::\_M\_transfer\_from\_if().

## 5.95.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

## 5.95.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_invalidate_↵
if (
    _Predicate __pred ) [inherited]
```

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

## 5.95.2.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.95.2.7 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.95.2.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_transfer_↵
from_if (
    _Safe_sequence< basic_string< _CharT, _Traits, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe\_sequence.tcc.

## 5.95.2.9 append() [1/3]

```
basic_string< _CharT, _Traits, _Allocator > & std::basic_string< _CharT, _Traits, _Allocator >_↵
::append (
    const basic_string< _CharT, _Traits, _Allocator > & __str ) [inherited]
```

Append a string to this string.



**Parameters**

<code>__str</code>	The string to append.
--------------------	-----------------------

**Returns**

Reference to this string.

Definition at line 775 of file `basic_string.tcc`.

**5.95.2.10 `append()`** [2/3]

```
basic_string<_CharT, _Traits, _Allocator> & std::basic_string<_CharT, _Traits, _Allocator>↵↵::append (↵    const basic_string<_CharT, _Traits, _Allocator> & __str,↵    size_type __pos,↵    size_type __n = npos ) [inherited]
```

Append a substring.

**Parameters**

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 792 of file `basic_string.tcc`.

**5.95.2.11 `append()`** [3/3]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (↵    initializer_list<_CharT> __l ) [inline], [inherited]
```

Append an `initializer_list` of characters.

## Parameters

<code>↩</code>	The initializer_list of characters to append.
<code>↩</code>	
<code>↩</code>	
<code>↩</code>	
<code>/</code>	

## Returns

Reference to this string.

Definition at line 4202 of file `basic_string.h`.

5.95.2.12 `assign()` [1/3]

```
basic_string<_CharT, _Traits, _Allocator> & std::basic_string<_CharT, _Traits, _Allocator>::assign (
    const basic_string<_CharT, _Traits, _Allocator> & __str ) [inherited]
```

Set value to contents of another string.

## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

## Returns

Reference to this string.

Definition at line 693 of file `basic_string.tcc`.

5.95.2.13 `assign()` [2/3]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (
    basic_string<_CharT, _Traits, _Allocator> && __str ) [inline], [inherited]
```

Set value to contents of another string.

## Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 4285 of file `basic_string.h`.

**5.95.2.14 `assign()`** [3/3]

```
basic_string& std::basic_string< _CharT, _Traits, _Allocator >::assign (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos,
    size_type __n = npos ) [inline], [inherited]
```

Set value to a substring of a string.

**Parameters**

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 4306 of file `basic_string.h`.

**5.95.2.15 `at()`** [1/2]

```
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
    size_type __n ) const [inline], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

<code>__↔ __n</code>	The index of the character to access.
--------------------------	---------------------------------------

**Returns**

Read-only (const) reference to the character.

**Exceptions**

<code>std::out_of_range</code>	If <i>n</i> is an invalid index.
--------------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 4006 of file `basic_string.h`.

**5.95.2.16 `at()`** [2/2]

```
reference std::basic_string<_CharT, _Traits, _Allocator>::at (  
    size_type __n ) [inline], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

<code>__↔ __n</code>	The index of the character to access.
--------------------------	---------------------------------------

**Returns**

Read/write reference to the character.

**Exceptions**

<code>std::out_of_range</code>	If <i>n</i> is an invalid index.
--------------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 4028 of file `basic_string.h`.

**5.95.2.17 back()** [1/2]

reference `std::basic_string<_CharT, _Traits, _Allocator>::back ( )` [inline], [inherited]

Returns a read/write reference to the data at the last element of the string.

Definition at line 4067 of file `basic_string.h`.

**5.95.2.18 back()** [2/2]

const\_reference `std::basic_string<_CharT, _Traits, _Allocator>::back ( ) const` [inline], [noexcept], [inherited]

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 4078 of file `basic_string.h`.

**5.95.2.19 capacity()**

size\_type `std::basic_string<_CharT, _Traits, _Allocator>::capacity ( ) const` [inline], [noexcept], [inherited]

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 3902 of file `basic_string.h`.

**5.95.2.20 compare()** [1/3]

int `std::basic_string<_CharT, _Traits, _Allocator>::compare ( const basic_string<_CharT, _Traits, _Allocator> & __str ) const` [inline], [inherited]

Compare to a string.

**Parameters**

<code>__str</code>	String to compare against.
--------------------	----------------------------

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The

function then compares the two strings by calling `traits::compare(data(), str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 5688 of file `basic_string.h`.

#### 5.95.2.21 `compare()` [2/3]

```
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Allocator> & __str ) const [inherited]
```

Compare substring to a string.

##### Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

##### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1385 of file `basic_string.tcc`.

#### 5.95.2.22 `compare()` [3/3]

```
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos2,
    size_type __n2 = npos ) const [inherited]
```

Compare substring to a substring.

##### Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of str.
<code>__n2</code>	Number of characters in substring of str.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1400 of file `basic_string.tcc`.

**5.95.2.23 empty()**

```
bool std::basic_string<_CharT, _Traits, _Allocator >::empty ( ) const [inline], [noexcept],  
[inherited]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 3952 of file `basic_string.h`.

**5.95.2.24 erase()** [1/2]

```
iterator std::basic_string<_CharT, _Traits, _Allocator >::erase (   
    iterator __position ) [inline], [inherited]
```

Remove one character.

**Parameters**

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 4652 of file `basic_string.h`.

5.95.2.25 `erase()` [2/2]

```

basic_string<_CharT, _Traits, _Allocator>::iterator std::basic_string<_CharT, _Traits, _
Allocator>::erase (
    iterator __first,
    iterator __last ) [inherited]

```

Remove a range of characters.

## Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

## Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 841 of file `basic_string.tcc`.

5.95.2.26 `find()`

```

size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = 0 ) const [inline], [noexcept], [inherited]

```

Find position of a string.

## Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5196 of file `basic_string.h`.



#### 5.95.2.27 find\_first\_not\_of()

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = 0 ) const    [inline], [noexcept], [inherited]
```

Find position of a character not in string.

##### Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5504 of file `basic_string.h`.

#### 5.95.2.28 find\_first\_of()

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
    const basic_string< _CharT, _Traits, _Allocator > & __str,
    size_type __pos = 0 ) const    [inline], [noexcept], [inherited]
```

Find position of a character of string.

##### Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5337 of file `basic_string.h`.

5.95.2.29 `find_last_not_of()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = npos ) const [inline], [noexcept], [inherited]
```

Find last position of a character not in string.

## Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5586 of file `basic_string.h`.

5.95.2.30 `find_last_of()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = npos ) const [inline], [noexcept], [inherited]
```

Find last position of a character of string.

## Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5421 of file `basic_string.h`.

**5.95.2.31 front()** [1/2]

reference `std::basic_string< _CharT, _Traits, _Allocator >::front ( )` [inline], [inherited]

Returns a read/write reference to the data at the first element of the string.

Definition at line 4045 of file `basic_string.h`.

**5.95.2.32 front()** [2/2]

const\_reference `std::basic_string< _CharT, _Traits, _Allocator >::front ( ) const` [inline], [noexcept], [inherited]

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 4056 of file `basic_string.h`.

**5.95.2.33 get\_allocator()**

allocator\_type `std::basic_string< _CharT, _Traits, _Allocator >::get_allocator ( ) const` [inline], [noexcept], [inherited]

Return copy of allocator used to construct this string.

Definition at line 5166 of file `basic_string.h`.

**5.95.2.34 insert()** [1/6]

```
void std::basic_string< _CharT, _Traits, _Allocator >::insert (
    iterator __p,
    size_type __n,
    _CharT __c ) [inline], [inherited]
```

Insert multiple characters.

**Parameters**

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4423 of file `basic_string.h`.

5.95.2.35 `insert()` [2/6]

```
void std::basic_string<_CharT, _Traits, _Allocator>::insert (
    iterator __p,
    _InputIterator __beg,
    _InputIterator __end ) [inline], [inherited]
```

Insert a range of characters.

## Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4440 of file `basic_string.h`.

5.95.2.36 `insert()` [3/6]

```
void std::basic_string<_CharT, _Traits, _Allocator>::insert (
    iterator __p,
    initializer_list<_CharT> __l ) [inline], [inherited]
```

Insert an `initializer_list` of characters.

**Parameters**

<code>_↔ _p</code>	Iterator referencing location in string to insert at.
<code>_↔ _l</code>	The initializer_list of characters to insert.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 4451 of file `basic_string.h`.

**5.95.2.37 insert()** [4/6]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (  
    size_type __pos1,  
    const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline], [inherited]
```

Insert value of a string.

**Parameters**

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4471 of file `basic_string.h`.

**5.95.2.38 insert()** [5/6]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (  
    size_type __pos1,
```

```
const basic_string<_CharT, _Traits, _Allocator > & __str,
size_type __pos2,
size_type __n = npos ) [inline], [inherited]
```

Insert a substring.

#### Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in <code>str</code> to insert.
<code>__n</code>	Number of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> .

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4493 of file `basic_string.h`.

#### 5.95.2.39 `insert()` [6/6]

```
iterator std::basic_string<_CharT, _Traits, _Allocator >::insert (
    iterator __p,
    _CharT __c ) [inline], [inherited]
```

Insert one character.

#### Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4575 of file `basic_string.h`.

**5.95.2.40 length()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::length ( ) const [inline], [noexcept], [inherited]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3845 of file `basic_string.h`.

**5.95.2.41 max\_size()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::max_size ( ) const [inline], [noexcept], [inherited]
```

Returns the size() of the largest possible string.

Definition at line 3850 of file `basic_string.h`.

**5.95.2.42 operator+=( )**

```
basic_string& std::basic_string< _CharT, _Traits, _Allocator >::operator+= (
    const basic_string< _CharT, _Traits, _Allocator > & __str ) [inline], [inherited]
```

Append a string to this string.

**Parameters**

<code>__str</code>	The string to append.
--------------------	-----------------------

**Returns**

Reference to this string.

Definition at line 4092 of file `basic_string.h`.

**5.95.2.43 `replace()`** [1/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline], [inherited]
```

Replace characters with value from another string.

**Parameters**

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4706 of file `basic_string.h`.

**5.95.2.44 `replace()`** [2/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
    size_type __pos1,
    size_type __n1,
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos2,
    size_type __n2 = npos ) [inline], [inherited]
```

Replace characters with value from another string.



**Parameters**

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of str to use.
<code>__n2</code>	Number of characters from str to use.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4728 of file `basic_string.h`.

**5.95.2.45 replace()** [3/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
    iterator __i1,
    iterator __i2,
    const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline], [inherited]
```

Replace range of characters with string.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4815 of file `basic_string.h`.

**5.95.2.46 `replace()`** [4/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s,
    size_type __n ) [inline], [inherited]
```

Replace range of characters with C substring.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4834 of file `basic_string.h`.

5.95.2.47 `replace()` [5/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s ) [inline], [inherited]
```

Replace range of characters with C string.

## Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4855 of file `basic_string.h`.

5.95.2.48 `replace()` [6/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (
    iterator __i1,
    iterator __i2,
    size_type __n,
    _CharT __c ) [inline], [inherited]
```

Replace range of characters with multiple characters.

## Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4876 of file `basic_string.h`.

**5.95.2.49 `replace()`** [7/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
    iterator __i1,
    iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2 ) [inline], [inherited]
```

Replace range of characters with range.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4900 of file `basic_string.h`.

**5.95.2.50** `replace()` [8/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (
    iterator __i1,
    iterator __i2,
    initializer_list<_CharT > __l ) [inline], [inherited]
```

Replace range of characters with `initializer_list`.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4969 of file `basic_string.h`.

**5.95.2.51** `reserve()`

```
void std::basic_string<_CharT, _Traits, _Allocator >::reserve (
    size_type __res_arg = 0 ) [inherited]
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

**Exceptions**

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 952 of file `basic_string.tcc`.

#### 5.95.2.52 `rfind()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
    const basic_string<_CharT, _Traits, _Allocator> & __str,
    size_type __pos = npos ) const [inline], [noexcept], [inherited]
```

Find last position of a string.

##### Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

##### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5258 of file `basic_string.h`.

#### 5.95.2.53 `size()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::size ( ) const [inline], [noexcept],
[inherited]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3839 of file `basic_string.h`.

#### 5.95.2.54 `swap()`

```
void std::basic_string<_CharT, _Traits, _Allocator>::swap (
    basic_string<_CharT, _Traits, _Allocator> & __s ) [inherited]
```

Swap contents with another string.

**Parameters**

<code>_↔_s</code>	String to swap with.
-------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 969 of file `basic_string.tcc`.

**5.95.3 Member Data Documentation****5.95.3.1 `_M_const_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

**5.95.3.2 `_M_iterators`**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

**5.95.3.3 `_M_version`**

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

5.95.3.4 `npos`

```
const basic_string<_CharT, _Traits, _Allocator>::size_type std::basic_string<_CharT, _Traits,
_AAllocator>::npos [static], [inherited]
```

Value returned by various member functions when they fail.

Definition at line 3310 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

5.96 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

## Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp` &\_\_b)
- `template<typename _Result, typename _Addend>`  
`_Result operator()` (`const _Result` &\_\_x, `const _Addend` &\_\_y)

## Public Attributes

- `_BinOp` & `__binop`

## 5.96.1 Detailed Description

```
template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>
```

General reduction, using a binary operator.

Definition at line 335 of file `for_each_selectors.h`.

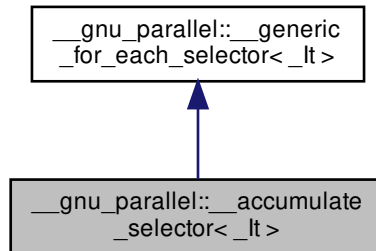
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)



### 5.97 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



#### Public Member Functions

- `template<typename _Op>`  
`std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

#### Public Attributes

- `_It _M_finish_iterator`

#### 5.97.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__accumulate_selector<_It>
```

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

#### 5.97.2 Member Function Documentation

##### 5.97.2.1 `operator()`

```
template<typename _It>
template<typename _Op>
std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator() (
    _Op __o,
    _It __i ) [inline]
```

Functor execution.

## Parameters

<code>_↔ _o</code>	Operator (unused).
<code>_↔ _i</code>	iterator referencing object.

## Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

## 5.97.3 Member Data Documentation

5.97.3.1 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

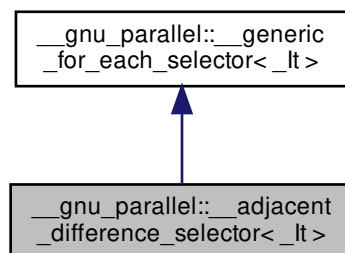
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.98 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



### Public Member Functions

- `template<typename _Op >  
bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It \_M\_finish\_iterator`

#### 5.98.1 Detailed Description

```
template<typename _It>  
struct __gnu_parallel::__adjacent_difference_selector< _It >
```

Selector that returns the difference between two adjacent \_\_elements.

Definition at line 269 of file `for_each_selectors.h`.

#### 5.98.2 Member Data Documentation

##### 5.98.2.1 `_M_finish_iterator`

```
template<typename _It >  
_It \_\_gnu\_parallel::\_\_generic\_for\_each\_selector< _It >::__M_finish_iterator [inherited]
```

\_\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

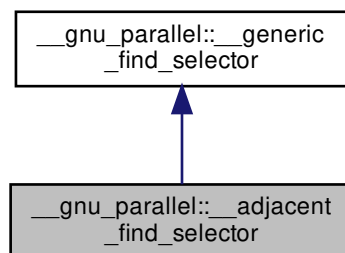
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 5.99 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >  
std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >  
bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## 5.99.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

## 5.99.2 Member Function Documentation

5.99.2.1 `_M_sequential_algorithm()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred ) [inline]
```

Corresponding sequential algorithm on a sequence.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 105 of file `find_selectors.h`.

5.99.2.2 `operator>()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred >
bool __gnu_parallel::__adjacent_find_selector::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred ) [inline]
```

Test on one position.

## Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

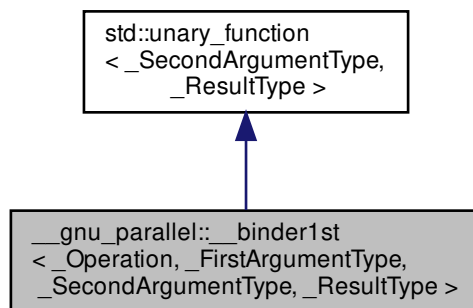
Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.100 `__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



## Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

## Public Member Functions

- `__binder1st` (`const _Operation &__x, const _FirstArgumentType &__y`)
- `_ResultType operator()` (`const _SecondArgumentType &__x`)
- `_ResultType operator()` (`_SecondArgumentType &__x`) `const`

#### Protected Attributes

- `_Operation` **`_M_op`**
- `_FirstArgumentType` **`_M_value`**

#### 5.100.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>  
class __gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder1st`, but giving the argument types explicitly.

Definition at line 192 of file `parallel/base.h`.

#### 5.100.2 Member Typedef Documentation

##### 5.100.2.1 `argument_type`

```
typedef _SecondArgumentType std::unary\_function< _SecondArgumentType , _ResultType >::argument\_type  
[inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

##### 5.100.2.2 `result_type`

```
typedef _ResultType std::unary\_function< _SecondArgumentType , _ResultType >::result\_type [inherited]
```

`result_type` is the return type

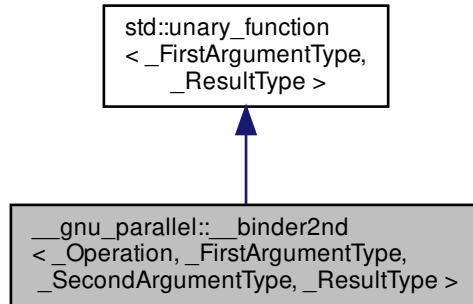
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.101 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



### Public Types

- typedef `_FirstArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

### Public Member Functions

- `__binder2nd` (`const _Operation &__x, const _SecondArgumentType &__y`)
- `_ResultType operator()` (`const _FirstArgumentType &__x`) `const`
- `_ResultType operator()` (`_FirstArgumentType &__x`)

### Protected Attributes

- `_Operation` `_M_op`
- `_SecondArgumentType` `_M_value`

#### 5.101.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder2nd`, but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

## 5.101.2 Member Typedef Documentation

## 5.101.2.1 argument\_type

```
typedef _FirstArgumentType std::unary_function< _FirstArgumentType , _ResultType >::argument_type  
[inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 5.101.2.2 result\_type

```
typedef _ResultType std::unary_function< _FirstArgumentType , _ResultType >::result_type [inherited]
```

result\_type is the return type

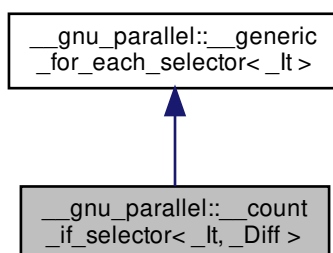
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.102 \_\_gnu\_parallel::\_\_count\_if\_selector&lt;\_It, \_Diff&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_count\_if\_selector<\_It, \_Diff>:





## Public Member Functions

- `template<typename _Op >`  
`_Diff operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 5.102.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_if_selector< _It, _Diff >
```

`std::count_if ()` selector.

Definition at line 194 of file `for_each_selectors.h`.

### 5.102.2 Member Function Documentation

#### 5.102.2.1 operator>()

```
template<typename _It, typename _Diff>
template<typename _Op >
_Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator() (
    _Op & __o,
    _It __i ) [inline]
```

Functor execution.

#### Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

#### Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

## 5.102.3 Member Data Documentation

5.102.3.1 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

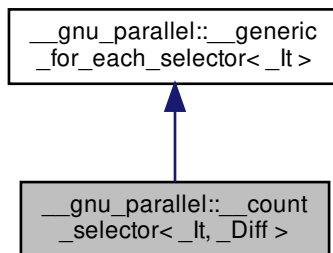
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.103 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:



## Public Member Functions

- `template<typename _ValueType >`  
`_Diff operator() (_ValueType &__v, _It __i)`

## Public Attributes

- `_It` [\\_M\\_finish\\_iterator](#)

### 5.103.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector< _It, _Diff >
```

std::count() selector.

Definition at line 180 of file `for_each_selectors.h`.

### 5.103.2 Member Function Documentation

#### 5.103.2.1 operator()()

```
template<typename _It, typename _Diff>
template<typename _ValueType >
_Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (
    _ValueType & __v,
    _It __i ) [inline]
```

Functor execution.

#### Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

#### Returns

1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

### 5.103.3 Member Data Documentation

#### 5.103.3.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

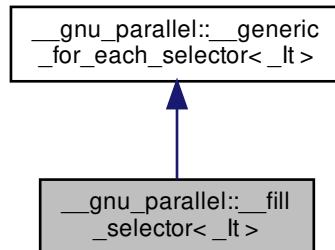
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.104 \_\_gnu\_parallel::\_\_fill\_selector&lt;\_It&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_fill\_selector<\_It>:



## Public Member Functions

- `template<typename _ValueType >`  
`bool operator() (_ValueType &__v, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 5.104.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__fill_selector<_It>
```

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

## 5.104.2 Member Function Documentation

## 5.104.2.1 operator&gt;()

```
template<typename _It >
template<typename _ValueType >
bool __gnu_parallel::__fill_selector<_It>::operator() (
    _ValueType & __v,
    _It __i ) [inline]
```

Functor execution.

## Parameters

<code>_↵ _v</code>	Current value.
<code>_↵ _i</code>	iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

## 5.104.3 Member Data Documentation

5.104.3.1 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

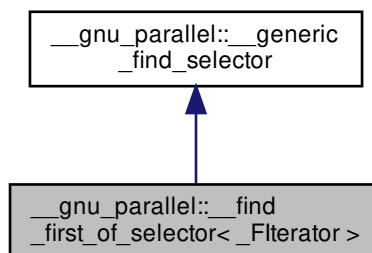
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.105 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



## Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair<_RAIter1, _RAIter2> __M_sequential_algorithm` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator()` (`_RAIter1 __i1`, `_RAIter2 __i2`, `_Pred __pred`)

## Public Attributes

- `_FIterator __M_begin`
- `_FIterator __M_end`

## 5.105.1 Detailed Description

```
template<typename _FIterator>
struct __gnu_parallel::__find_first_of_selector<_FIterator>
```

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

## 5.105.2 Member Function Documentation

5.105.2.1 `__M_sequential_algorithm()`

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred ) [inline]
```

Corresponding sequential algorithm on a sequence.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 186 of file find\_selectors.h.

References `std::make_pair()`.

### 5.105.2.2 `operator()`

```
template<typename _FIterator >
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
bool __gnu_parallel::__find_first_of_selector< _FIterator >::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred ) [inline]
```

Test on one position.

#### Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

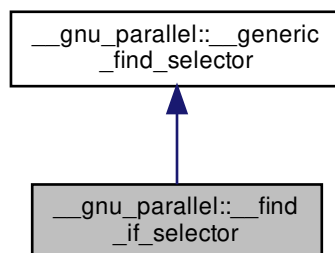
Definition at line 169 of file find\_selectors.h.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

### 5.106 `__gnu_parallel::__find_if_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair<_RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## 5.106.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

## 5.106.2 Member Function Documentation

5.106.2.1 `_M_sequential_algorithm()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_if_selector::_M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred ) [inline]
```

Corresponding sequential algorithm on a sequence.

## Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

5.106.2.2 `operator()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred >
bool __gnu_parallel::__find_if_selector::operator() (
```



```

    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred ) [inline]

```

Test on one position.

#### Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

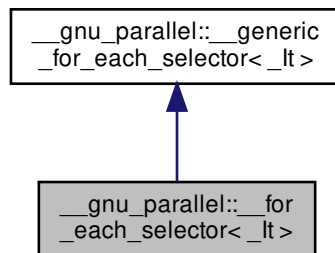
Definition at line 60 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

### 5.107 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



#### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

#### Public Attributes

- `_It` [\\_M\\_finish\\_iterator](#)

### 5.107.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__for_each_selector<_It>
```

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

### 5.107.2 Member Function Documentation

#### 5.107.2.1 `operator()`

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__for_each_selector<_It>::operator() (
    _Op & __o,
    _It __i ) [inline]
```

Functor execution.

#### Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

### 5.107.3 Member Data Documentation

#### 5.107.3.1 `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
```

Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

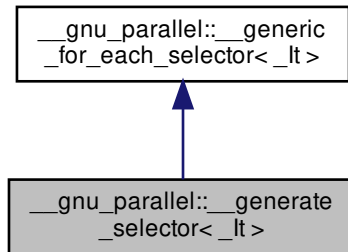
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.108 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.108.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generate_selector<_It>
```

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

#### 5.108.2 Member Function Documentation

##### 5.108.2.1 `operator()()`

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__generate_selector<_It>::operator() (
    _Op & __o,
    _It __i ) [inline]
```

Functor execution.

## Parameters

<a href="#"><code>_↵ _o</code></a>	Operator.
<a href="#"><code>_↵ _i</code></a>	iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

## 5.108.3 Member Data Documentation

## 5.108.3.1 \_M\_finish\_iterator

```
template<typename _It >  
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

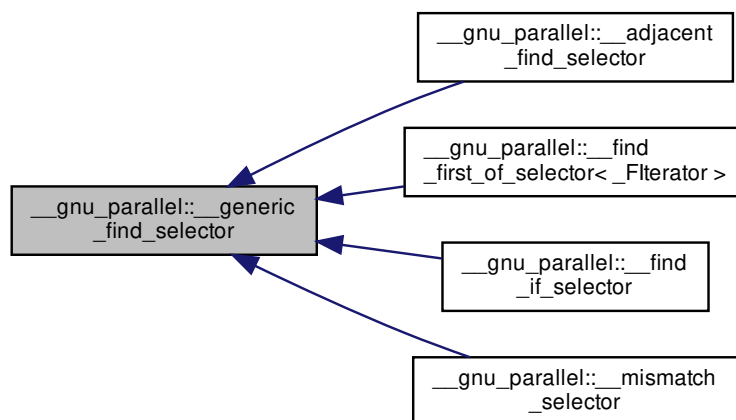
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.109 \_\_gnu\_parallel::\_\_generic\_find\_selector Struct Reference

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



### 5.109.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

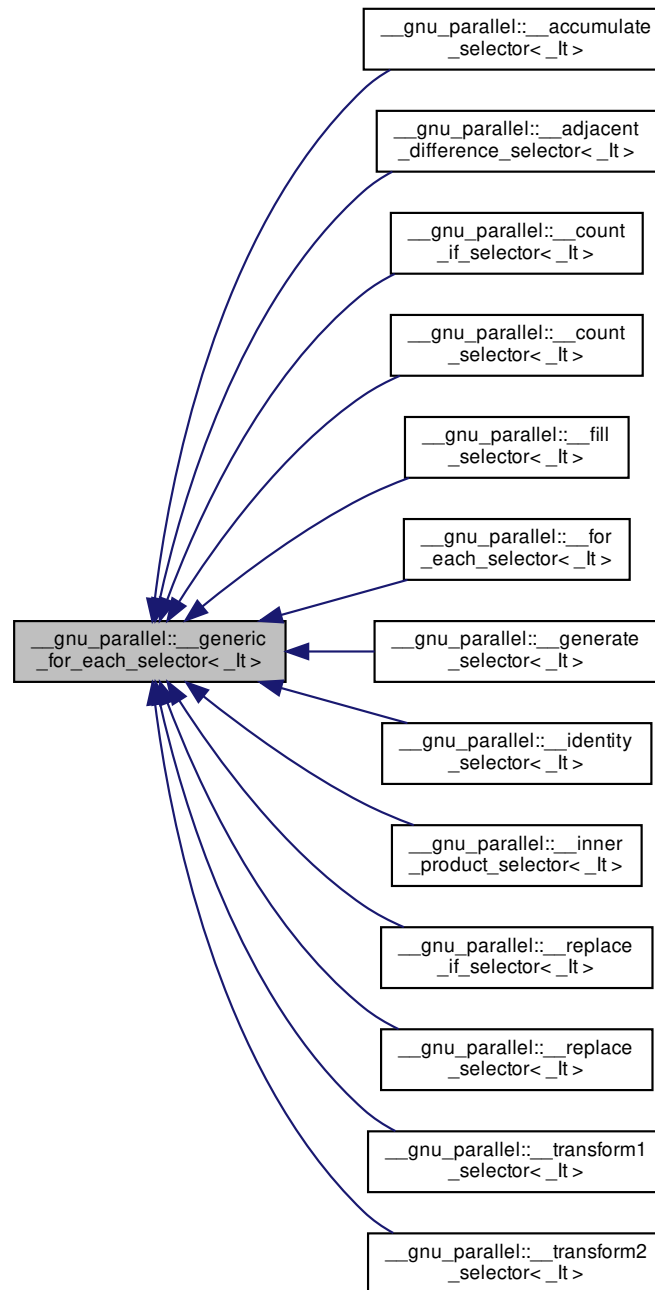
Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.110 \_\_gnu\_parallel::\_\_generic\_for\_each\_selector&lt;\_It&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>:



## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

## 5.110.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generic_for_each_selector< _It >
```

Generic \_\_selector for embarrassingly parallel functions.

Definition at line 42 of file `for_each_selectors.h`.

## 5.110.2 Member Data Documentation

## 5.110.2.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

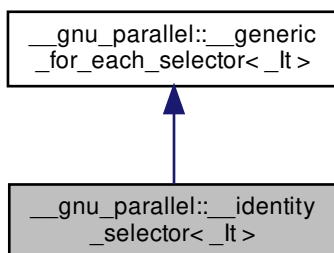
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.111 \_\_gnu\_parallel::\_\_identity\_selector&lt; \_It &gt; Struct Template Reference

Inheritance diagram for `__gnu_parallel::__identity_selector< _It >`:



## Public Member Functions

- `template<typename _Op >  
_It operator() (_Op __o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 5.111.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__identity_selector<_It>
```

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

### 5.111.2 Member Function Documentation

#### 5.111.2.1 operator>()

```
template<typename _It>
template<typename _Op >
_It __gnu_parallel::__identity_selector<_It>::operator() (
    _Op __o,
    _It __i ) [inline]
```

Functor execution.

#### Parameters

<code>↵ _o</code>	Operator (unused).
<code>↵ _i</code>	iterator referencing object.

#### Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.



### 5.111.3 Member Data Documentation

#### 5.111.3.1 \_M\_finish\_iterator

```
template<typename _It >
__It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

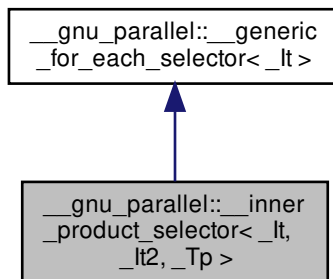
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.112 \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp > Struct Template Reference

Inheritance diagram for `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`:



#### Public Member Functions

- [\\_\\_inner\\_product\\_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op >`  
`_Tp operator()` (`_Op __mult, _It __current`)

#### Public Attributes

- `_It` [\\_\\_begin1\\_iterator](#)
- `_It2` [\\_\\_begin2\\_iterator](#)
- `_It` [\\_M\\_finish\\_iterator](#)

## 5.112.1 Detailed Description

```
template<typename _It, typename _It2, typename _Tp>
struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>
```

`std::inner_product()` selector.

Definition at line 222 of file `for_each_selectors.h`.

## 5.112.2 Constructor &amp; Destructor Documentation

5.112.2.1 `__inner_product_selector()`

```
template<typename _It , typename _It2 , typename _Tp >
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__inner_product_selector (
    _It __b1,
    _It2 __b2 ) [inline], [explicit]
```

Constructor.

## Parameters

<code>__b1</code>	Begin iterator of first sequence.
<code>__b2</code>	Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

## 5.112.3 Member Function Documentation

5.112.3.1 `operator()()`

```
template<typename _It , typename _It2 , typename _Tp >
template<typename _Op >
_Tp __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator() (
    _Op __mult,
    _It __current ) [inline]
```

Functor execution.

## Parameters

<code>__mult</code>	Multiplication functor.
<code>__current</code>	iterator referencing object.

## Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_iterator`, and `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin2_iterator`.

## 5.112.4 Member Data Documentation

### 5.112.4.1 `__begin1_iterator`

```
template<typename _It , typename _It2 , typename _Tp >
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator
```

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()()`.

### 5.112.4.2 `__begin2_iterator`

```
template<typename _It , typename _It2 , typename _Tp >
_It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator
```

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()()`.

### 5.112.4.3 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.113 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

## Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

## Public Attributes

- `_Compare & __comp`

## 5.113.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.114 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

## Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

## Public Attributes

- `_Compare & __comp`

## 5.114.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.

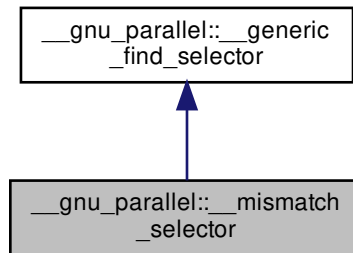
Definition at line 307 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.115 `__gnu_parallel::__mismatch_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__mismatch_selector`:



### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair<_RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

#### 5.115.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

#### 5.115.2 Member Function Documentation

##### 5.115.2.1 `_M_sequential_algorithm()`

```

template<typename _RAIter1, typename _RAIter2, typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (
    _RAIter1 __begin1,
    _RAIter1 __end1,
    _RAIter2 __begin2,
    _Pred __pred ) [inline]
  
```

Corresponding sequential algorithm on a sequence.

#### Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 143 of file `find_selectors.h`.

#### 5.115.2.2 `operator()`

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
bool __gnu_parallel::__mismatch_selector::operator() (
    _RAIter1 __i1,
    _RAIter2 __i2,
    _Pred __pred ) [inline]
```

Test on one position.

#### Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

#### 5.116 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

#### Public Member Functions

- `_RAIter3 operator()` (`_RAIterliterator __seqs_begin`, `_RAIterliterator __seqs_end`, `_RAIter3 __target`, `__DifferenceTp __length`, `_Compare __comp`)

#### 5.116.1 Detailed Description

```
template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare
>
```

Switch for 3-way merging with \_\_sentinels turned off.

Note that 3-way merging is always stable!

Definition at line 752 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 5.117 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `_DifferenceTp __length`, `_Compare __comp`)

#### 5.117.1 Detailed Description

```
template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with \_\_sentinels turned on.

Note that 3-way merging is always stable!

Definition at line 772 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 5.118 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `_DifferenceTp __length`, `_Compare __comp`)

#### 5.118.1 Detailed Description

```
template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare
>
```

Switch for 4-way merging with `__sentinels` turned off.

Note that 4-way merging is always stable!

Definition at line 795 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 5.119 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _↵ DifferenceTp, _Compare > Struct Template Reference`

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `_↵ DifferenceTp __length`, `_Compare __comp`)

#### 5.119.1 Detailed Description

```
template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

Definition at line 815 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 5.120 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference`

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `const type- name std::iterator_traits< typename std::iterator_traits< _RAIterlterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp`)



### 5.120.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp,
_Compare >
```

Switch for k-way merging with \_\_sentinels turned on.

Definition at line 837 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.121 \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< false, \_\_stable, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare > Struct Template Reference

#### Public Member Functions

- **\_RAIter3 operator()** (\_RAIterIterator \_\_seqs\_begin, \_RAIterIterator \_\_seqs\_end, \_RAIter3 \_\_target, const typename std::iterator\_traits< typename std::iterator\_traits< \_RAIterIterator >::value\_type::first\_type >::value\_type &\_\_sentinel, \_DifferenceTp \_\_length, \_Compare \_\_comp)

### 5.121.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with \_\_sentinels turned off.

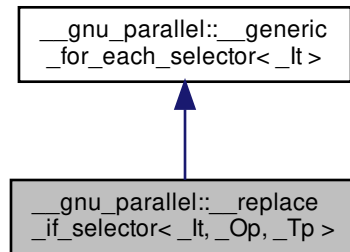
Definition at line 872 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.122 \_\_gnu\_parallel::\_\_replace\_if\_selector&lt; \_It, \_Op, \_Tp &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_replace\_if\_selector< \_It, \_Op, \_Tp >:



## Public Member Functions

- [\\_\\_replace\\_if\\_selector](#) (const \_Tp & [\\_\\_new\\_val](#))
- bool [operator\(\)](#) (\_Op &\_\_o, \_It \_\_i)

## Public Attributes

- const \_Tp & [\\_\\_new\\_val](#)
- \_It [\\_M\\_finish\\_iterator](#)

## 5.122.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >
```

std::replace() selector.

Definition at line 156 of file `for_each_selectors.h`.

## 5.122.2 Constructor &amp; Destructor Documentation

## 5.122.2.1 \_\_replace\_if\_selector()

```
template<typename _It, typename _Op, typename _Tp>
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (
    const _Tp & __new_val ) [inline], [explicit]
```

Constructor.

**Parameters**

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 164 of file `for_each_selectors.h`.

**5.122.3 Member Function Documentation****5.122.3.1 operator()**

```
template<typename _It, typename _Op, typename _Tp>
bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
    _Op & __o,
    _It __i ) [inline]
```

Functor execution.

**Parameters**

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val`.

**5.122.4 Member Data Documentation****5.122.4.1 \_\_new\_val**

```
template<typename _It, typename _Op, typename _Tp>
const _Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val
```

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator()()`.

5.122.4.2 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector<_It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

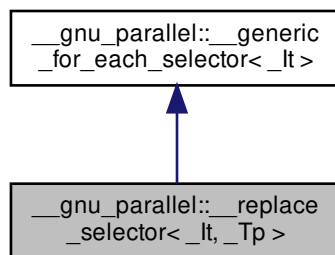
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.123 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



## Public Member Functions

- [\\_\\_replace\\_selector](#) (const `_Tp` & `__new_val`)
- [bool operator\(\)](#) (`_Tp` & `__v`, `_It` `__i`)

## Public Attributes

- const `_Tp` & [\\_\\_new\\_val](#)
- `_It` [\\_M\\_finish\\_iterator](#)

## 5.123.1 Detailed Description

```
template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector<_It, _Tp>
```

`std::replace()` selector.

Definition at line 132 of file `for_each_selectors.h`.

### 5.123.2 Constructor & Destructor Documentation

#### 5.123.2.1 \_\_replace\_selector()

```
template<typename _It , typename _Tp >
__gnu_parallel::__replace_selector< _It, _Tp >::__replace_selector (
    const _Tp & __new_val ) [inline], [explicit]
```

Constructor.

##### Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 140 of file `for_each_selectors.h`.

### 5.123.3 Member Function Documentation

#### 5.123.3.1 operator()()

```
template<typename _It , typename _Tp >
bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (
    _Tp & __v,
    _It __i ) [inline]
```

Functor execution.

##### Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Definition at line 146 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_selector< _It, _Tp >::__new_val`.

### 5.123.4 Member Data Documentation

## 5.123.4.1 \_\_new\_val

```
template<typename _It , typename _Tp >
const _Tp& __gnu_parallel::__replace_selector<_It, _Tp >::__new_val
```

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp >::operator()`.

## 5.123.4.2 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector<_It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

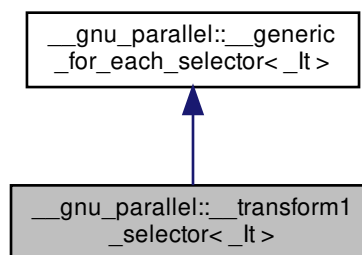
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.124 \_\_gnu\_parallel::\_\_transform1\_selector&lt;\_It&gt; Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



## Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &__o, _It __i)`

## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

### 5.124.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform1_selector< _It >
```

std::transform() \_\_selector, one input sequence variant.

Definition at line 100 of file for\_each\_selectors.h.

### 5.124.2 Member Function Documentation

#### 5.124.2.1 operator>()

```
template<typename _It>
template<typename _Op >
bool __gnu_parallel::__transform1_selector< _It >::operator() (
    _Op & __o,
    _It __i ) [inline]
```

Functor execution.

#### Parameters

<a href="#">↩</a> <code>__o</code>	Operator.
<a href="#">↩</a> <code>__i</code>	iterator referencing object.

Definition at line 107 of file for\_each\_selectors.h.

### 5.124.3 Member Data Documentation

#### 5.124.3.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

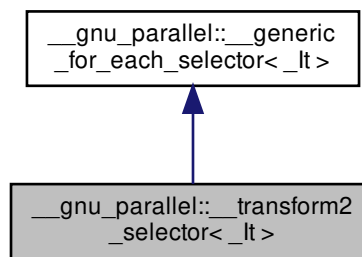
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.125 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:



### Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 5.125.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform2_selector<_It>
```

`std::transform()` `__selector`, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.





## 5.125.2 Member Function Documentation

### 5.125.2.1 operator{ }()

```
template<typename _It>
template<typename _Op >
bool __gnu_parallel::__transform2_selector< _It >::operator() (
    _Op & __o,
    _It __i ) [inline]
```

Functor execution.

#### Parameters

<a href="#">_o</a>	Operator.
<a href="#">_i</a>	iterator referencing object.

Definition at line 123 of file `_each_selectors.h`.

## 5.125.3 Member Data Documentation

### 5.125.3.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

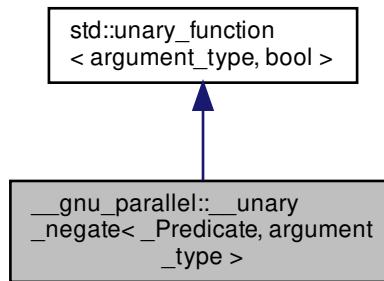
Definition at line 47 of file `_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.126 `__gnu_parallel::__unary_negate<_Predicate, argument_type>` Class Template Reference

Inheritance diagram for `__gnu_parallel::__unary_negate<_Predicate, argument_type>`:



## Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

## Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- `bool operator()` (`const argument_type &__x`)

## Protected Attributes

- `_Predicate _M_pred`

## 5.126.1 Detailed Description

```
template<typename _Predicate, typename argument_type>  
class __gnu_parallel::__unary_negate<_Predicate, argument_type>
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file `parallel/base.h`.

## 5.126.2 Member Typedef Documentation

#### 5.126.2.1 `argument_type`

```
typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.126.2.2 `result_type`

```
typedef bool std::unary_function< argument_type , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

### 5.127 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

#### Public Types

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef std::iterator\_traits< `_RAIter` > **`_TraitsType`**
- typedef `_TraitsType::value_type` **`_ValueType`**

#### Public Member Functions

- [`\_DRandomShufflingGlobalData`](#) (`_RAIter` &\_\_source)

#### Public Attributes

- `_ThreadIndex` \* `_M_bin_proc`
- `_DifferenceType` \*\* `_M_dist`
- int `_M_num_bins`
- int `_M_num_bits`
- `_RAIter` & `_M_source`
- `_DifferenceType` \* `_M_starts`
- `_ValueType` \*\* `_M_temporaries`

## 5.127.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>
```

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 52 of file `random_shuffle.h`.

## 5.127.2 Constructor &amp; Destructor Documentation

5.127.2.1 `_DRandomShufflingGlobalData()`

```
template<typename _RAIter>
__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_DRandomShufflingGlobalData (
    _RAIter & __source ) [inline]
```

Constructor.

Definition at line 83 of file `random_shuffle.h`.

## 5.127.3 Member Data Documentation

5.127.3.1 `_M_bin_proc`

```
template<typename _RAIter>
_ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

5.127.3.2 `_M_dist`

```
template<typename _RAIter>
_DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist
```

Two-dimensional array to hold the thread-bin distribution.

Dimensions `(_M_num_threads + 1) __x (_M_num_bins + 1)`.

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs←_pu()`.

### 5.127.3.3 `_M_num_bins`

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins
```

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

### 5.127.3.4 `_M_num_bits`

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits
```

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

### 5.127.3.5 `_M_source`

```
template<typename _RAIter>
_RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source
```

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

### 5.127.3.6 `_M_starts`

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts
```

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

5.127.3.7 `_M_temporaries`

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_temporaries
```

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

5.128 `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

## Public Attributes

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData<\\_RAIter> \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

## 5.128.1 Detailed Description

```
template<typename _RAIter, typename _RandomNumberGenerator>
struct __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>
```

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 91 of file `random_shuffle.h`.

## 5.128.2 Member Data Documentation

5.128.2.1 `__bins_end`

```
template<typename _RAIter, typename _RandomNumberGenerator>
_BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::__bins_end
```

End index for bins taken care of by this thread.

Definition at line 100 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

#### 5.128.2.2 `_M_bins_begin`

```
template<typename _RAIter, typename _RandomNumberGenerator>
_binIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin
```

Begin index for bins taken care of by this thread.

Definition at line 97 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

#### 5.128.2.3 `_M_num_threads`

```
template<typename _RAIter, typename _RandomNumberGenerator>
int __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads
```

Number of threads participating in total.

Definition at line 94 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

#### 5.128.2.4 `_M_sd`

```
template<typename _RAIter, typename _RandomNumberGenerator>
_DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumber↵
Generator >::_M_sd
```

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

#### 5.128.2.5 `_M_seed`

```
template<typename _RAIter, typename _RandomNumberGenerator>
uint32_t __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed
```

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

5.129 `__gnu_parallel::_DummyReduct` Struct Reference

## Public Member Functions

- `bool operator()` (`bool`, `bool`) `const`

## 5.129.1 Detailed Description

Reduction function doing nothing.

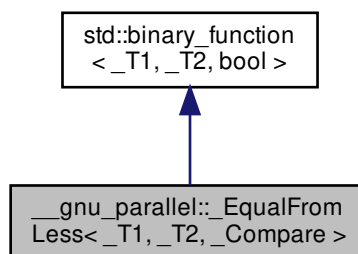
Definition at line 298 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

5.130 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>`:



## Public Types

- `typedef _T1` [first\\_argument\\_type](#)
- `typedef bool` [result\\_type](#)
- `typedef _T2` [second\\_argument\\_type](#)

## Public Member Functions

- `_EqualFromLess` (`_Compare` & `__comp`)
- `bool operator()` (`const _T1` & `__a`, `const _T2` & `__b`)



### 5.130.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>  
class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file parallel/base.h.

### 5.130.2 Member Typedef Documentation

#### 5.130.2.1 first\_argument\_type

```
typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.130.2.2 result\_type

```
typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.130.2.3 second\_argument\_type

```
typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

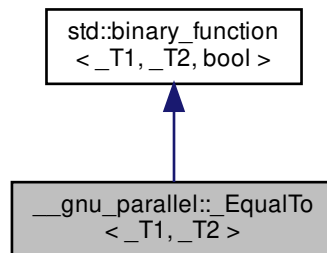
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.131 \_\_gnu\_parallel::\_EqualTo&lt;\_T1, \_T2&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_EqualTo<\_T1, \_T2>:



#### Public Types

- typedef `_T1` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_T2` `second_argument_type`

#### Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

##### 5.131.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_EqualTo<_T1, _T2>
```

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `parallel/base.h`.

##### 5.131.2 Member Typedef Documentation

#### 5.131.2.1 first\_argument\_type

```
typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.131.2.2 result\_type

```
typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.131.2.3 second\_argument\_type

```
typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

### 5.132 \_\_gnu\_parallel::\_GuardedIterator< \_RAIter, \_Compare > Class Template Reference

#### Public Member Functions

- [\\_GuardedIterator](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Compare](#) &\_\_comp)
- [operator \\_RAIter](#) ()
- [std::iterator\\_traits< \\_RAIter >::value\\_type](#) & [operator\\*](#) ()
- [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > & [operator++](#) ()

#### Friends

- bool [operator](#)< ([\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > &\_\_bi1, [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > &\_\_bi2)
- bool [operator](#)<= ([\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > &\_\_bi1, [\\_GuardedIterator](#)< [\\_RAIter](#), [\\_Compare](#) > &\_\_bi2)

## 5.132.1 Detailed Description

```
template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator<_RAIter, _Compare>
```

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 73 of file `multiway_merge.h`.

## 5.132.2 Constructor &amp; Destructor Documentation

5.132.2.1 `_GuardedIterator()`

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator<_RAIter, _Compare>::_GuardedIterator (
    _RAIter __begin,
    _RAIter __end,
    _Compare & __comp ) [inline]
```

Constructor. Sets iterator to beginning of sequence.

## Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator provided for associated overloaded compare operators.

Definition at line 91 of file `multiway_merge.h`.

## 5.132.3 Member Function Documentation

5.132.3.1 `operator _RAIter()`

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator _RAIter ( ) [inline]
```

Convert to wrapped iterator.

**Returns**

Wrapped iterator.

Definition at line 112 of file multiway\_merge.h.

**5.132.3.2 operator\*()**

```
template<typename _RAIter, typename _Compare>
std::iterator_traits<_RAIter>::value_type& __gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator* ( ) [inline]
```

Dereference operator.

**Returns**

Referenced element.

Definition at line 107 of file multiway\_merge.h.

**5.132.3.3 operator++()**

```
template<typename _RAIter, typename _Compare>
__GuardedIterator<_RAIter, _Compare>& __gnu_parallel::_GuardedIterator< _RAIter, _Compare >↔
::operator++ ( ) [inline]
```

Pre-increment operator.

**Returns**

This.

Definition at line 98 of file multiway\_merge.h.

**5.132.4 Friends And Related Function Documentation****5.132.4.1 operator<**

```
template<typename _RAIter, typename _Compare>
bool operator< (
    __GuardedIterator< _RAIter, _Compare > & __bi1,
    __GuardedIterator< _RAIter, _Compare > & __bi2 ) [friend]
```

Compare two elements referenced by guarded iterators.

## Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

## Returns

true if less.

Definition at line 120 of file `multiway_merge.h`.

5.132.4.2 `operator<=`

```
template<typename _RAIter, typename _Compare>
bool operator<= (
    _GuardedIterator< _RAIter, _Compare > & __bi1,
    _GuardedIterator< _RAIter, _Compare > & __bi2 ) [friend]
```

Compare two elements referenced by guarded iterators.

## Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

## Returns

True if less equal.

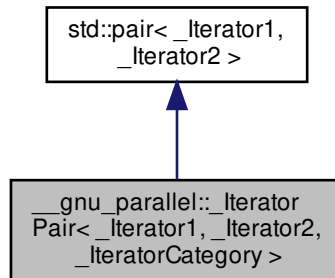
Definition at line 135 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

### 5.133 `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>`:



#### Public Types

- using `_PCCFP` = `_PCC<lis_same<_Iterator1, _U1 >::value||lis_same<_Iterator2, _U2 >::value, _Iterator1, _Iterator2 >`
- using `_PCCP` = `_PCC< true, _Iterator1, _Iterator2 >`
- typedef `std::iterator_traits<_Iterator1 >_TraitsType`
- typedef `_TraitsType::difference_type` **`difference_type`**
- typedef `_Iterator1` **`first_type`**
- typedef `_IteratorCategory` **`iterator_category`**
- typedef `_IteratorPair` \* **`pointer`**
- typedef `_IteratorPair` & **`reference`**
- typedef `_Iterator2` **`second_type`**
- typedef void **`value_type`**

#### Public Member Functions

- **`_IteratorPair`** (`const _Iterator1` &\_\_first, `const _Iterator2` &\_\_second)
- **`operator _Iterator2`** () const
- **`_IteratorPair`** **`operator+`** (`difference_type` \_\_delta) const
- **`_IteratorPair`** & **`operator++`** ()
- const **`_IteratorPair`** **`operator++`** (int)
- `difference_type` **`operator-`** (const **`_IteratorPair`** &\_\_other) const
- **`_IteratorPair`** & **`operator--`** ()
- const **`_IteratorPair`** **`operator--`** (int)
- **`_IteratorPair`** & **`operator=`** (const **`_IteratorPair`** &\_\_other)
- void **`swap`** (**`pair`** &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Iterator1 >, \_\_is\_nothrow\_swappable<\_Iterator2 >>::value)

#### Public Attributes

- `_Iterator1` [first](#)
- `_Iterator2` [second](#)

#### 5.133.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

#### 5.133.2 Member Typedef Documentation

##### 5.133.2.1 `_PCCFP`

```
using std::pair<_Iterator1, _Iterator2>::_PCCFP = _PCC<!is_same<_Iterator1, _U1>::value ||
!is_same<_Iterator2, _U2>::value, _Iterator1, _Iterator2> [inherited]
```

There is also a templated copy ctor for the `pair` class itself.

Definition at line 283 of file `stl_pair.h`.

##### 5.133.2.2 `_PCCP`

```
using std::pair<_Iterator1, _Iterator2>::_PCCP = _PCC<true, _Iterator1, _Iterator2> [inherited]
```

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 252 of file `stl_pair.h`.

##### 5.133.2.3 `second_type`

```
typedef _Iterator2 std::pair<_Iterator1, _Iterator2>::second_type [inherited]
```

`first_type` is the first bound type

Definition at line 212 of file `stl_pair.h`.



### 5.133.3 Member Data Documentation

#### 5.133.3.1 first

`_Iterator1` [std::pair](#)< `_Iterator1` , `_Iterator2` >::first [inherited]

`second_type` is the second bound type

Definition at line 214 of file `stl_pair.h`.

#### 5.133.3.2 second

`_Iterator2` [std::pair](#)< `_Iterator1` , `_Iterator2` >::second [inherited]

`first` is a copy of the first object

Definition at line 215 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

### 5.134 `__gnu_parallel::IteratorTriple`< `_Iterator1`, `_Iterator2`, `_Iterator3`, `_IteratorCategory` > Class Template Reference

#### Public Types

- typedef `std::iterator_traits`< `_Iterator1` >::difference\_type **difference\_type**
- typedef `_IteratorCategory` **iterator\_category**
- typedef [\\_IteratorTriple](#) \* **pointer**
- typedef [\\_IteratorTriple](#) & **reference**
- typedef void **value\_type**

#### Public Member Functions

- **\_IteratorTriple** (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second, const `_Iterator3` &\_\_third)
- **operator \_Iterator3** () const
- [\\_IteratorTriple](#) **operator+** (difference\_type \_\_delta) const
- [\\_IteratorTriple](#) & **operator++** ()
- const [\\_IteratorTriple](#) **operator++** (int)
- difference\_type **operator-** (const [\\_IteratorTriple](#) &\_\_other) const
- [\\_IteratorTriple](#) & **operator--** ()
- const [\\_IteratorTriple](#) **operator--** (int)
- [\\_IteratorTriple](#) & **operator=** (const [\\_IteratorTriple](#) &\_\_other)

#### Public Attributes

- `_Iterator1` **`_M_first`**
- `_Iterator2` **`_M_second`**
- `_Iterator3` **`_M_third`**

##### 5.134.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>  
class __gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory >
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

## 5.135 \_\_gnu\_parallel::\_Job<\_DifferenceTp> Struct Template Reference

#### Public Types

- `typedef _DifferenceTp` **`_DifferenceType`**

#### Public Attributes

- `volatile _DifferenceType` [\\_M\\_first](#)
- `volatile _DifferenceType` [\\_M\\_last](#)
- `volatile _DifferenceType` [\\_M\\_load](#)

##### 5.135.1 Detailed Description

```
template<typename _DifferenceTp>  
struct __gnu_parallel::_Job<_DifferenceTp >
```

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

##### 5.135.2 Member Data Documentation

#### 5.135.2.1 `_M_first`

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_first
```

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file workstealing.h.

#### 5.135.2.2 `_M_last`

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_last
```

Last element.

Changed by owning thread only.

Definition at line 67 of file workstealing.h.

#### 5.135.2.3 `_M_load`

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load
```

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

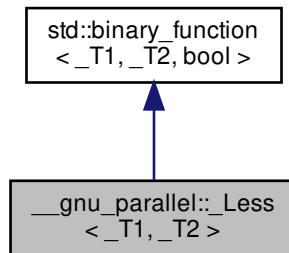
Definition at line 72 of file workstealing.h.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

5.136 `__gnu_parallel::_Less<_T1, _T2>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



## Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

## Public Member Functions

- `bool` **operator()** (`const _T1 &__t1, const _T2 &__t2`) `const`
- `bool` **operator()** (`const _T2 &__t2, const _T1 &__t1`) `const`

## 5.136.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_Less<_T1, _T2>
```

Similar to `std::less`, but allows two different types.

Definition at line 252 of file `parallel/base.h`.

## 5.136.2 Member Typedef Documentation

### 5.136.2.1 first\_argument\_type

```
typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

### 5.136.2.2 result\_type

```
typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

### 5.136.2.3 second\_argument\_type

```
typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

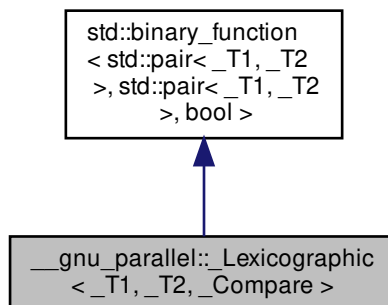
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.137 \_\_gnu\_parallel::\_Lexicographic< \_T1, \_T2, \_Compare > Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_Lexicographic< \_T1, \_T2, \_Compare >:



## Public Types

- typedef `std::pair< _T1, _T2 >` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair< _T1, _T2 >` `second_argument_type`

## Public Member Functions

- **`_Lexicographic`** (`_Compare &__comp`)
- `bool operator()` (`const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2`) `const`

## 5.137.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >
```

Compare \_\_a pair of types lexicographically, ascending.

Definition at line 53 of file `multiseq_selection.h`.

## 5.137.2 Member Typedef Documentation

5.137.2.1 `first_argument_type`

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

5.137.2.2 `result_type`

```
typedef bool std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >↵
::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

### 5.137.2.3 second\_argument\_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

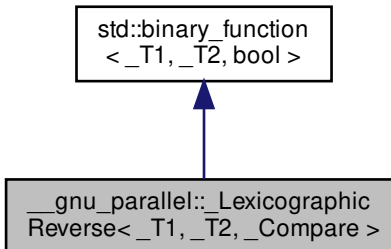
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

## 5.138 \_\_gnu\_parallel::\_LexicographicReverse< \_T1, \_T2, \_Compare > Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_LexicographicReverse< \_T1, \_T2, \_Compare >:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `_LexicographicReverse` (`_Compare &__comp`)
- `bool operator()` (`const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2`) `const`

### 5.138.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>
```

Compare \_\_a pair of types lexicographically, descending.

Definition at line 80 of file `multiseq_selection.h`.

### 5.138.2 Member Typedef Documentation

#### 5.138.2.1 `first_argument_type`

```
typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.138.2.2 `result_type`

```
typedef bool std::binary_function<_T1, _T2, bool>::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 5.138.2.3 `second_argument_type`

```
typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

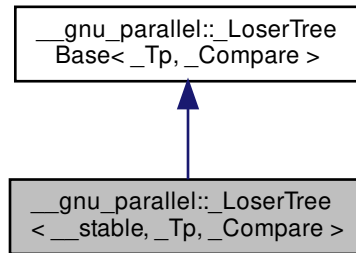
The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)



### 5.139 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



#### Public Member Functions

- **`_LoserTree`** (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

#### Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_log_k`
- unsigned int `_M_offset`

#### 5.139.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >
```

Stable `_LoserTree` variant.

Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 169 of file `losertree.h`.

## 5.139.2 Member Function Documentation

5.139.2.1 `__delete_min_insert()`

```
template<bool __stable, typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
    _Tp __key,
    bool __sup ) [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 222 of file `losertree.h`.

5.139.2.2 `__get_min_source()`

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ( ) [inline], [inherited]
```

**Returns**

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

5.139.2.3 `__insert_start()`

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
    const _Tp & __key,
    int __source,
    bool __sup ) [inline], [inherited]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

**Parameters**

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

### 5.139.3 Member Data Documentation

#### 5.139.3.1 `_M_log_k`

```
template<typename _Tp , typename _Compare >
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]
```

`log_2{_M_k}`

Definition at line 72 of file losertree.h.

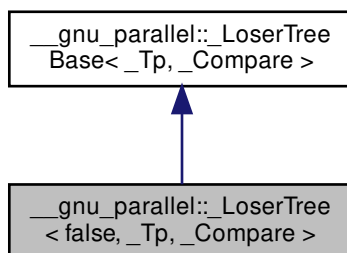
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.140 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



## Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

## 5.140.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< false, _Tp, _Compare >
```

Unstable `_LoserTree` variant.

Stability (non-stable here) is selected with partial specialization.

Definition at line 261 of file `losertree.h`.

## 5.140.2 Member Function Documentation

5.140.2.1 `__delete_min_insert()`

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert (
    _Tp __key,
    bool __sup ) [inline]
```

Delete the `_M_key` smallest element and insert the element `__key` instead.

## Parameters

<code>__key</code>	the <code>_M_key</code> to insert
<code>__sup</code>	true iff <code>__key</code> is an explicitly marked supremum

Definition at line 324 of file `losertree.h`.

#### 5.140.2.2 `__get_min_source()`

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ( ) [inline], [inherited]
```

##### Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

#### 5.140.2.3 `__init_winner()`

```
template<typename _Tp , typename _Compare >
unsigned int __gnu_parallel::_LoserTreeBase< false, _Tp, _Compare >::__init_winner (
    unsigned int __root ) [inline]
```

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

##### Parameters

<code>__root</code>	<code>__index</code> of the "game" to start.
---------------------	--

Definition at line 284 of file losertree.h.

#### 5.140.2.4 `__insert_start()`

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
    const _Tp & __key,
    int __source,
    bool __sup ) [inline], [inherited]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

##### Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> <code>__sequence</code>
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

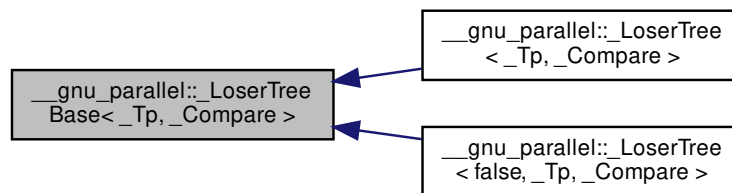
References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_first_insert`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_sup`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.141 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>`:



### Classes

- [struct `\_Loser`](#)

### Public Member Functions

- [\\_LoserTreeBase](#) (unsigned int \_\_k, \_Compare \_\_comp)
- [~\\_LoserTreeBase](#) ()
- [int \\_\\_get\\_min\\_source](#) ()
- [void \\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

### Protected Attributes

- [\\_Compare \\_M\\_comp](#)
- [bool \\_M\\_first\\_insert](#)
- [unsigned int \\_M\\_ik](#)
- [unsigned int \\_M\\_k](#)
- [unsigned int \\_M\\_log\\_k](#)
- [\\_Loser \\* \\_M\\_losers](#)
- [unsigned int \\_M\\_offset](#)

### 5.141.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

#### Parameters

<code>_Tp</code>	the element type
<code>_Compare</code>	the comparator to use, defaults to <code>std::less&lt;_Tp&gt;</code>

Definition at line 55 of file `losertree.h`.

### 5.141.2 Constructor & Destructor Documentation

#### 5.141.2.1 `_LoserTreeBase()`

```
template<typename _Tp , typename _Compare >
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (
    unsigned int __k,
    _Compare __comp ) [inline]
```

The constructor.

#### Parameters

<code>__k</code>	The number of sequences to merge.
<code>__comp</code>	The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::_rd_log2()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

## 5.141.2.2 ~\_LoserTreeBase()

```
template<typename _Tp , typename _Compare >
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase ( ) [inline]
```

The destructor.

Definition at line 118 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

## 5.141.3 Member Function Documentation

## 5.141.3.1 \_\_get\_min\_source()

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ( ) [inline]
```

## Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

## 5.141.3.2 \_\_insert\_start()

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
    const _Tp & __key,
    int __source,
    bool __sup ) [inline]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

## Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .



Definition at line 134 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

#### 5.141.4 Member Data Documentation

##### 5.141.4.1 `_M_comp`

```
template<typename _Tp , typename _Compare >
_Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected]
```

`_Compare` to use.

Definition at line 78 of file losertree.h.

##### 5.141.4.2 `_M_first_insert`

```
template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected]
```

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

##### 5.141.4.3 `_M_log_k`

```
template<typename _Tp , typename _Compare >
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected]
```

`log_2[_M_k]`

Definition at line 72 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.141.4.4 `_M_losers`

```
template<typename _Tp, typename _Compare>
_Loser* __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_losers [protected]
```

`_LoserTree` \_\_elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__get_min_source()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__insert_start()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.142 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser` Struct Reference

## Public Attributes

- `_Tp` [\\_M\\_key](#)
- `int` [\\_M\\_source](#)
- `bool` [\\_M\\_sup](#)

## 5.142.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser
```

Internal representation of a `_LoserTree` element.

Definition at line 59 of file `losertree.h`.

## 5.142.2 Member Data Documentation

5.142.2.1 `_M_key`

```
template<typename _Tp, typename _Compare>
_Tp __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_key
```

`_M_key` of the element in the `_LoserTree`.

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__insert_start()`.

### 5.142.2.2 `_M_source`

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source
```

\_\_index of the \_\_source \_\_sequence.

Definition at line 64 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

### 5.142.2.3 `_M_sup`

```
template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup
```

flag, true iff this is a "maximum" \_\_sentinel.

Definition at line 62 of file losertree.h.

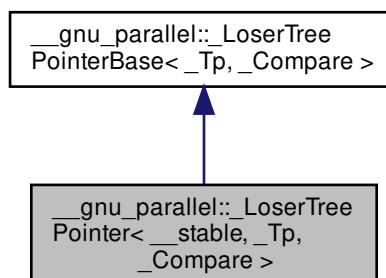
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

## 5.143 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



## Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void `__delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

## 5.143.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `_LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.

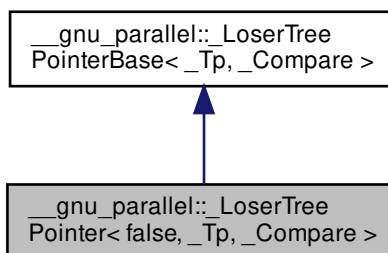
Definition at line 409 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.144 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTreePointer** (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

## 5.144.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable \_LoserTree implementation.

The stable variant is above.

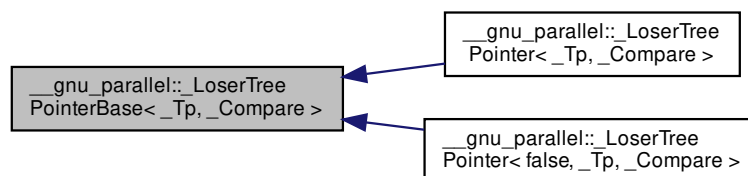
Definition at line 491 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.145 \_\_gnu\_parallel::\_LoserTreePointerBase&lt; \_Tp, \_Compare &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreePointerBase< \_Tp, \_Compare >:



## Classes

- struct [\\_Loser](#)

## Public Member Functions

- **`_LoserTreePointerBase`** (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)<\_Tp>())
- int **`__get_min_source`** ()
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- \_Compare **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- [\\_Loser](#) \* **`_M_losers`**
- unsigned int **`_M_offset`**

## 5.145.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare >
```

Base class of `_Loser` Tree implementation using pointers.

Definition at line 357 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.146 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>::_Loser` Struct Reference

## Public Attributes

- const \_Tp \* **`_M_keyp`**
- int **`_M_source`**
- bool **`_M_sup`**

### 5.146.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser
```

Internal representation of `_LoserTree` \_\_elements.

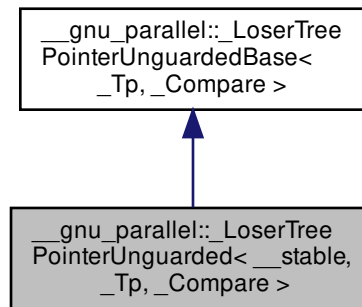
Definition at line 361 of file `losertree.h`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

### 5.147 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



#### Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **`__delete_min_insert`** (const \_Tp &\_\_key, bool \_\_sup)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int \_\_root)
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- unsigned int **`_M_ik`**
- unsigned int **`_M_offset`**

### 5.147.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >
```

Stable unguarded `_LoserTree` variant storing pointers.

Unstable variant is implemented below using partial specialization.

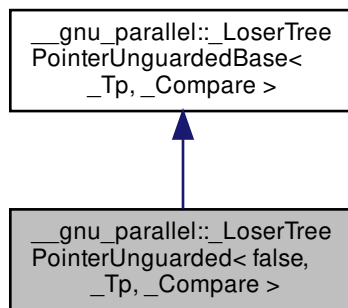
Definition at line 891 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.148 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



### Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int `__k`, const `_Tp` & `__sentinel`, `_Compare` `__comp`=[std::less](#)< `_Tp` >())
- void **`__delete_min_insert`** (const `_Tp` & `__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` & `__key`, int `__source`, bool)



## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

## 5.148.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
```

Unstable unguarded `_LoserTree` variant storing pointers.

Stable variant is above.

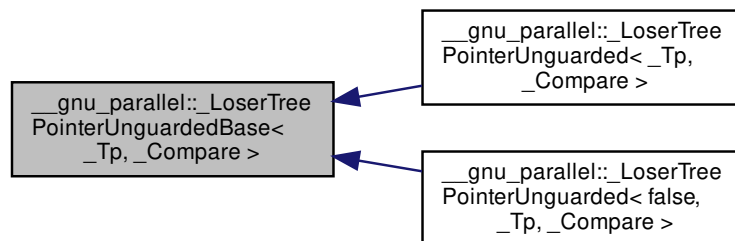
Definition at line 977 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.149 `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTreePointerUnguardedBase** (unsigned int `__k`, const `_Tp &__sentinel`, `_Compare __comp=std::less<_Tp>()`)
- int **\_\_get\_min\_source** ()
- void **\_\_insert\_start** (const `_Tp &__key`, int `__source`, bool)

## Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

## 5.149.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 828 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.150 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

## Static Public Attributes

- static const bool `_M_use_pointer`

## 5.150.1 Detailed Description

```
template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits<_Tp>
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };

template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

#### Parameters

<code>_Tp</code>	type to give the loser tree traits for.
------------------	---

Definition at line 731 of file `multiway_merge.h`.

### 5.150.2 Member Data Documentation

#### 5.150.2.1 `_M_use_pointer`

```
template<typename _Tp >
const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer [static]
```

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

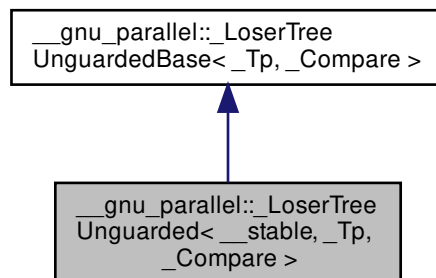
Definition at line 739 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 5.151 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



## Public Member Functions

- `_LoserTreeUnguarded` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void `__delete_min_insert` (\_Tp \_\_key, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

## 5.151.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded `_LoserTree`.

Unstable variant is selected below with partial specialization.

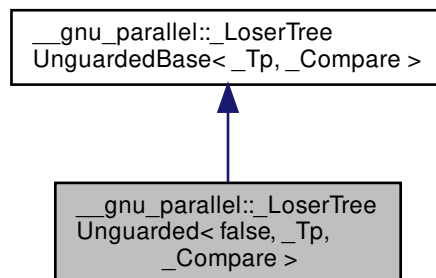
Definition at line 646 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.152 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTreeUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

## 5.152.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
```

Non-Stable implementation of unguarded \_LoserTree.

Stable implementation is above.

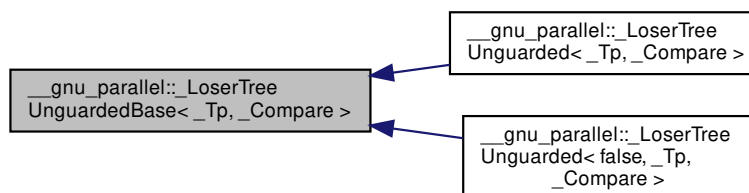
Definition at line 734 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.153 \_\_gnu\_parallel::\_LoserTreeUnguardedBase&lt; \_Tp, \_Compare &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreeUnguardedBase< \_Tp, \_Compare >:



## Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less<\\_Tp>\(\)](#))
- `int __get_min_source` ()
- `void __insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

## 5.153.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>
```

Base class for unguarded `_LoserTree` implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

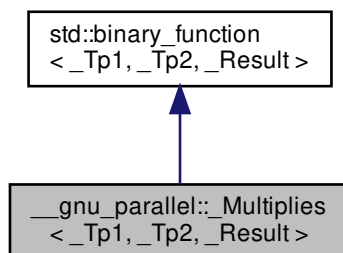
Definition at line 574 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.154 `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>`:



## Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result` **operator()** (const `_Tp1` &\_\_x, const `_Tp2` &\_\_y) const

### 5.154.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ (*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))>
struct __gnu_parallel::Multiplies< _Tp1, _Tp2, _Result >
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `parallel/base.h`.

### 5.154.2 Member Typedef Documentation

#### 5.154.2.1 first\_argument\_type

```
typedef _Tp1 std::binary\_function< _Tp1 , _Tp2 , _Result >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.154.2.2 result\_type

```
typedef _Result std::binary\_function< _Tp1 , _Tp2 , _Result >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

## 5.154.2.3 second\_argument\_type

```
typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.155 \_\_gnu\_parallel::\_Nothing Struct Reference

## Public Member Functions

- `template<typename _It >  
void operator\(\) (_It __i)`

## 5.155.1 Detailed Description

Functor doing nothing.

For some \_\_reduction tasks (this is not a function object, but is passed as \_\_selector \_\_dummy parameter.

Definition at line 288 of file for\_each\_selectors.h.






## 5.155.2 Member Function Documentation

## 5.155.2.1 operator&gt;()

```
template<typename _It >  
void __gnu_parallel::_Nothing::operator() (   
    _It __i ) [inline]
```

Functor execution.

## Parameters

	iterator referencing object.
	
	
	
	



Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.156 `__gnu_parallel::_Piece<_DifferenceTp >` Struct Template Reference

### Public Types

- `typedef _DifferenceTp _DifferenceType`

### Public Attributes

- `_DifferenceType \_M\_begin`
- `_DifferenceType \_M\_end`

#### 5.156.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::_Piece< _DifferenceTp >
```

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

#### 5.156.2 Member Data Documentation

##### 5.156.2.1 `_M_begin`

```
template<typename _DifferenceTp >
_DifferenceType \_\_gnu\_parallel::\_Piece< \_DifferenceTp >::\_M\_begin
```

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

5.156.2.2 `_M_end`

```
template<typename _DifferenceTp >
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end
```

End of subsequence.

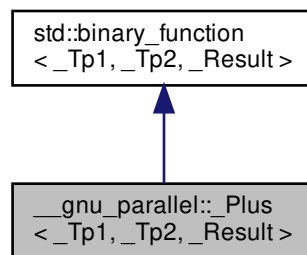
Definition at line 54 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

5.157 `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result>`:



## Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result` **operator()** (const `_Tp1` &`__x`, const `_Tp2` &`__y`) const

## 5.157.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))>
struct __gnu_parallel::_Plus<_Tp1, _Tp2, _Result>
```

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `parallel/base.h`.

## 5.157.2 Member Typedef Documentation

### 5.157.2.1 first\_argument\_type

```
typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

### 5.157.2.2 result\_type

```
typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

### 5.157.2.3 second\_argument\_type

```
typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.158 \_\_gnu\_parallel::\_PMWSSortingData< \_RAIter > Struct Template Reference

### Public Types

- typedef \_TraitsType::difference\_type **\_DifferenceType**
- typedef std::iterator\_traits< \_RAIter > **\_TraitsType**
- typedef \_TraitsType::value\_type **\_ValueType**

## Public Attributes

- [\\_ThreadIndex](#) [\\_M\\_num\\_threads](#)
- [\\_DifferenceType](#) \* [\\_M\\_offsets](#)
- [std::vector](#)< [\\_Piece](#)< [\\_DifferenceType](#) > > \* [\\_M\\_pieces](#)
- [\\_ValueType](#) \* [\\_M\\_samples](#)
- [\\_RAIter](#) [\\_M\\_source](#)
- [\\_DifferenceType](#) \* [\\_M\\_starts](#)
- [\\_ValueType](#) \*\* [\\_M\\_temporary](#)

## 5.158.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_PMWMSSortingData< _RAIter >
```

Data accessed by all threads.

PMWMS = parallel multiway mergesort

Definition at line 61 of file multiway\_mergesort.h.

## 5.158.2 Member Data Documentation

## 5.158.2.1 \_M\_num\_threads

```
template<typename _RAIter>
\_ThreadIndex \_\_gnu\_parallel::\_PMWMSSortingData< \_RAIter >::_M_num_threads
```

Number of threads involved.

Definition at line 68 of file multiway\_mergesort.h.

Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

## 5.158.2.2 \_M\_offsets

```
template<typename _RAIter>
\_DifferenceType* \_\_gnu\_parallel::\_PMWMSSortingData< \_RAIter >::_M_offsets
```

Offsets to add to the found positions.

Definition at line 83 of file multiway\_mergesort.h.

#### 5.158.2.3 `_M_pieces`

```
template<typename _RAIter>
std::vector<_Piece<_DifferenceType> >* __gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces
```

Pieces of data to merge [thread][\_\_sequence].

Definition at line 86 of file multiway\_mergesort.h.

#### 5.158.2.4 `_M_samples`

```
template<typename _RAIter>
_ValueType* __gnu_parallel::_PMWMSortingData< _RAIter >::_M_samples
```

Samples.

Definition at line 80 of file multiway\_mergesort.h.

Referenced by `__gnu_parallel::_determine_samples()`.

#### 5.158.2.5 `_M_source`

```
template<typename _RAIter>
_RAIter __gnu_parallel::_PMWMSortingData< _RAIter >::_M_source
```

Input \_\_begin.

Definition at line 71 of file multiway\_mergesort.h.

Referenced by `__gnu_parallel::_determine_samples()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 5.158.2.6 `_M_starts`

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSortingData< _RAIter >::_M_starts
```

Start indices, per thread.

Definition at line 74 of file multiway\_mergesort.h.

Referenced by `__gnu_parallel::_determine_samples()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

5.158.2.7 `_M_temporary`

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_PMWMSortingData<_RAIter>::_M_temporary
```

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

5.159 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

## Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**
- typedef `_PseudoSequenceIterator<_Tp, uint64_t>` **`iterator`**

## Public Member Functions

- `_PseudoSequence` (`const _Tp &__val, _DifferenceType __count`)
- `iterator begin` () const
- `iterator end` () const

## 5.159.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

## Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

## 5.159.2 Constructor & Destructor Documentation

### 5.159.2.1 `_PseudoSequence()`

```
template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::_PseudoSequence (
    const _Tp & __val,
    _DifferenceType __count ) [inline]
```

Constructor.

#### Parameters

<code>__val</code>	Element of the sequence.
<code>__count</code>	Number of (virtual) copies.

Definition at line 371 of file `parallel/base.h`.

## 5.159.3 Member Function Documentation

### 5.159.3.1 `begin()`

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin ( ) const [inline]
```

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

### 5.159.3.2 `end()`

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end ( ) const [inline]
```

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.160 `__gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>` Class Template Reference

## Public Types

- `typedef _DifferenceTp _DifferenceType`

## Public Member Functions

- `_PseudoSequenceliterator (const _Tp &__val, _DifferenceType __pos)`
- `bool operator!= (const \_PseudoSequenceliterator &__i2)`
- `const _Tp & operator* () const`
- `\_PseudoSequenceliterator & operator++ ()`
- `\_PseudoSequenceliterator operator++ (int)`
- `_DifferenceType operator- (const \_PseudoSequenceliterator &__i2)`
- `bool operator== (const \_PseudoSequenceliterator &__i2)`
- `const _Tp & operator[] (_DifferenceType) const`

## 5.160.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>
```

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

## Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.161 `__gnu_parallel::_QSBThreadLocal<_RAIter>` Struct Template Reference

## Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::pair<_RAIter, _RAIter> _Piece`
- `typedef std::iterator\_traits<_RAIter> _TraitsType`



## Public Member Functions

- [\\_QSBThreadLocal](#) (int \_\_queue\_size)

## Public Attributes

- volatile [\\_DifferenceType](#) \* [\\_M\\_elements\\_leftover](#)
- [\\_Piece](#) [\\_M\\_global](#)
- [\\_Piece](#) [\\_M\\_initial](#)
- [\\_RestrictedBoundedConcurrentQueue](#)< [\\_Piece](#) > [\\_M\\_leftover\\_parts](#)
- [\\_ThreadIndex](#) [\\_M\\_num\\_threads](#)

### 5.161.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_QSBThreadLocal< _RAIter >
```

Information local to one thread in the parallel quicksort run.

Definition at line 65 of file `balanced_quicksort.h`.

### 5.161.2 Member Typedef Documentation

#### 5.161.2.1 [\\_Piece](#)

```
template<typename _RAIter>
typedef std::pair<_RAIter, _RAIter> \_\_gnu\_parallel::\_QSBThreadLocal< _RAIter >::\_Piece
```

Continuous part of the sequence, described by an iterator pair.

Definition at line 72 of file `balanced_quicksort.h`.

### 5.161.3 Constructor & Destructor Documentation

#### 5.161.3.1 [\\_QSBThreadLocal](#)()

```
template<typename _RAIter>
\_\_gnu\_parallel::\_QSBThreadLocal< _RAIter >::\_QSBThreadLocal (
    int __queue_size ) [inline]
```

Constructor.

## Parameters

<code>__queue_size</code>	size of the work-stealing queue.
---------------------------	----------------------------------

Definition at line 91 of file `balanced_quicksort.h`.

## 5.161.4 Member Data Documentation

5.161.4.1 `_M_elements_leftover`

```
template<typename _RAIter>
volatile _DifferenceType* __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_elements_leftover
```

Pointer to a counter of elements left over to sort.

Definition at line 84 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__parallel_sort_qsb()`.

5.161.4.2 `_M_global`

```
template<typename _RAIter>
_Piece __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_global
```

The complete sequence to sort.

Definition at line 87 of file `balanced_quicksort.h`.

5.161.4.3 `_M_initial`

```
template<typename _RAIter>
_Piece __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_initial
```

Initial piece to work on.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

#### 5.161.4.4 `_M_leftover_parts`

```
template<typename _RAIter>
_RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_↵
leftover_parts
```

Work-stealing queue.

Definition at line 78 of file `balanced_quicksort.h`.

#### 5.161.4.5 `_M_num_threads`

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads
```

Number of threads involved in this algorithm.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

### 5.162 `__gnu_parallel::_RandomNumber` Class Reference

#### Public Member Functions

- [\\_RandomNumber](#) ()
- [\\_RandomNumber](#) (uint32\_t \_\_seed, uint64\_t \_M\_supremum=0x100000000ULL)
- unsigned long [\\_\\_genrand\\_bits](#) (int \_\_bits)
- uint32\_t [operator\(\)](#) ()
- uint32\_t [operator\(\)](#) (uint64\_t local\_supremum)

#### 5.162.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

#### 5.162.2 Constructor & Destructor Documentation

## 5.162.2.1 \_RandomNumber() [1/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber ( ) [inline]
```

Default constructor. Seed with 0.

Definition at line 74 of file random\_number.h.

## 5.162.2.2 \_RandomNumber() [2/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber (
    uint32_t __seed,
    uint64_t _M_supremum = 0x100000000ULL ) [inline]
```

Constructor.

## Parameters

<code>__seed</code>	Random <code>__seed</code> .
<code>_M_supremum</code>	Generate integer random numbers in the interval <code>[0, _M_supremum)</code> .

Definition at line 85 of file random\_number.h.

## 5.162.3 Member Function Documentation

## 5.162.3.1 \_\_genrand\_bits()

```
unsigned long __gnu_parallel::_RandomNumber::__genrand_bits (
    int __bits ) [inline]
```

Generate a number of random bits, run-time parameter.

## Parameters

<code>__bits</code>	Number of bits to generate.
---------------------	-----------------------------

Definition at line 109 of file random\_number.h.

## 5.162.3.2 operator()() [1/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() ( ) [inline]
```

Generate unsigned random 32-bit integer.

Definition at line 94 of file random\_number.h.

### 5.162.3.3 operator() [2/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() (
    uint64_t local_supremum ) [inline]
```

Generate unsigned random 32-bit integer in the interval [0,local\_supremum).

Definition at line 100 of file random\_number.h.

The documentation for this class was generated from the following file:

- [random\\_number.h](#)

## 5.163 \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue<\_Tp> Class Template Reference

### Public Member Functions

- [\\_RestrictedBoundedConcurrentQueue](#) ([\\_SequenceIndex](#) \_\_max\_size)
- [~\\_RestrictedBoundedConcurrentQueue](#) ()
- bool [pop\\_back](#) ([\\_Tp](#) &\_\_t)
- bool [pop\\_front](#) ([\\_Tp](#) &\_\_t)
- void [push\\_front](#) (const [\\_Tp](#) &\_\_t)

### 5.163.1 Detailed Description

```
template<typename _Tp>
class __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>
```

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

#### Parameters

<code>_Tp</code>	Contained element type.
------------------	-------------------------

Definition at line 52 of file queue.h.

## 5.163.2 Constructor &amp; Destructor Documentation

## 5.163.2.1 \_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue (
    __SequenceIndex __max_size ) [inline]
```

Constructor. Not to be called concurrent, of course.

## Parameters

<code>__max_size</code>	Maximal number of elements to be contained.
-------------------------	---

Definition at line 68 of file queue.h.

## 5.163.2.2 ~\_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::~~_RestrictedBoundedConcurrentQueue ( )
[inline]
```

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

## 5.163.3 Member Function Documentation

## 5.163.3.1 pop\_back()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (
    _Tp & __t ) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop\_front().

Definition at line 127 of file queue.h.

### 5.163.3.2 pop\_front()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (
    _Tp & __t ) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop\_front().

Definition at line 100 of file queue.h.

### 5.163.3.3 push\_front()

```
template<typename _Tp>
void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (
    const _Tp & __t ) [inline]
```

Pushes one element into the queue at the front end. Must not be called concurrently with pop\_front().

Definition at line 83 of file queue.h.

The documentation for this class was generated from the following file:

- [queue.h](#)

## 5.164 \_\_gnu\_parallel::\_SamplingSorter< \_\_stable, \_RAIter, \_StrictWeakOrdering > Struct Template Reference

### Public Member Functions

- void **operator()** (\_RAIter \_\_first, \_RAIter \_\_last, \_StrictWeakOrdering \_\_comp)

### 5.164.1 Detailed Description

```
template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
```

Stable sorting functor.

Used to reduce code instantiation in multiway\_merge\_sampling\_splitting.

Definition at line 1007 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

5.165 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

## Public Member Functions

- void **operator()** ( \_RAIter \_\_first, \_RAIter \_\_last, \_StrictWeakOrdering \_\_comp)

## 5.165.1 Detailed Description

```
template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >
```

Non-\_\_stable sorting functor.

Used to reduce code instantiation in multiway\_merge\_sampling\_splitting.

Definition at line 1020 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

5.166 `__gnu_parallel::_Settings` Struct Reference

## Static Public Member Functions

- static const [\\_Settings](#) & [get](#) () throw ()
- static void [set](#) ( [\\_Settings](#) &) throw ()

## Public Attributes

- [\\_SequenceIndex](#) [accumulate\\_minimal\\_n](#)
- unsigned int [adjacent\\_difference\\_minimal\\_n](#)
- [\\_AlgorithmStrategy](#) [algorithm\\_strategy](#)
- unsigned int [cache\\_line\\_size](#)
- [\\_SequenceIndex](#) [count\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [fill\\_minimal\\_n](#)
- [\\_FindAlgorithm](#) [find\\_algorithm](#)
- double [find\\_increasing\\_factor](#)
- [\\_SequenceIndex](#) [find\\_initial\\_block\\_size](#)
- [\\_SequenceIndex](#) [find\\_maximum\\_block\\_size](#)
- float [find\\_scale\\_factor](#)
- [\\_SequenceIndex](#) [find\\_sequential\\_search\\_size](#)
- [\\_SequenceIndex](#) [for\\_each\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [generate\\_minimal\\_n](#)
- unsigned long long [L1\\_cache\\_size](#)
- unsigned long long [L2\\_cache\\_size](#)
- [\\_SequenceIndex](#) [max\\_element\\_minimal\\_n](#)



- [\\_SequenceIndex](#) `merge_minimal_n`
- unsigned int `merge_oversampling`
- [\\_SplittingAlgorithm](#) **`merge_splitting`**
- [\\_SequenceIndex](#) `min_element_minimal_n`
- [\\_MultiwayMergeAlgorithm](#) **`multiway_merge_algorithm`**
- int `multiway_merge_minimal_k`
- [\\_SequenceIndex](#) `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- [\\_SplittingAlgorithm](#) **`multiway_merge_splitting`**
- [\\_SequenceIndex](#) `nth_element_minimal_n`
- [\\_SequenceIndex](#) `partial_sort_minimal_n`
- [\\_PartialSumAlgorithm](#) **`partial_sum_algorithm`**
- float `partial_sum_dilation`
- unsigned int `partial_sum_minimal_n`
- double `partition_chunk_share`
- [\\_SequenceIndex](#) `partition_chunk_size`
- [\\_SequenceIndex](#) `partition_minimal_n`
- [\\_SequenceIndex](#) `qsb_steals`
- unsigned int `random_shuffle_minimal_n`
- [\\_SequenceIndex](#) `replace_minimal_n`
- [\\_SequenceIndex](#) `search_minimal_n`
- [\\_SequenceIndex](#) `set_difference_minimal_n`
- [\\_SequenceIndex](#) `set_intersection_minimal_n`
- [\\_SequenceIndex](#) `set_symmetric_difference_minimal_n`
- [\\_SequenceIndex](#) `set_union_minimal_n`
- [\\_SortAlgorithm](#) **`sort_algorithm`**
- [\\_SequenceIndex](#) `sort_minimal_n`
- unsigned int `sort_mwms_oversampling`
- unsigned int `sort_qs_num_samples_preset`
- [\\_SequenceIndex](#) `sort_qsb_base_case_maximal_n`
- [\\_SplittingAlgorithm](#) **`sort_splitting`**
- unsigned int `TLB_size`
- [\\_SequenceIndex](#) `transform_minimal_n`
- [\\_SequenceIndex](#) `unique_copy_minimal_n`
- [\\_SequenceIndex](#) **`workstealing_chunk_size`**

#### 5.166.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file `settings.h`.

#### 5.166.2 Member Function Documentation

### 5.166.2.1 get()

```
static const \_Settings& __gnu_parallel::_Settings::get ( ) throw ( ) [static]
```

Get the global settings.

Referenced by `__gnu_parallel::find_template()`, `__gnu_parallel::parallel_nth_element()`, `__gnu_parallel::parallel_partial_sum()`, `__gnu_parallel::parallel_partial_sum_linear()`, `__gnu_parallel::parallel_partition()`, `__gnu_parallel::parallel_random_shuffle_drs()`, `__gnu_parallel::parallel_sort()`, `__gnu_parallel::qsb_local_sort_with_helping()`, `__gnu_parallel::sequential_random_shuffle()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

### 5.166.2.2 set()

```
static void __gnu_parallel::_Settings::set (
    \_Settings & ) throw ( ) [static]
```

Set the global settings.

## 5.166.3 Member Data Documentation

### 5.166.3.1 accumulate\_minimal\_n

```
\_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n
```

Minimal input size for accumulate.

Definition at line 139 of file settings.h.

### 5.166.3.2 adjacent\_difference\_minimal\_n

```
unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n
```

Minimal input size for adjacent\_difference.

Definition at line 142 of file settings.h.

#### 5.166.3.3 `cache_line_size`

```
unsigned int __gnu_parallel::_Settings::cache_line_size
```

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Definition at line 265 of file settings.h.

#### 5.166.3.4 `count_minimal_n`

```
\_SequenceIndex __gnu_parallel::_Settings::count_minimal_n
```

Minimal input size for count and count\_if.

Definition at line 145 of file settings.h.

#### 5.166.3.5 `fill_minimal_n`

```
\_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n
```

Minimal input size for fill.

Definition at line 148 of file settings.h.

#### 5.166.3.6 `find_increasing_factor`

```
double __gnu_parallel::_Settings::find_increasing_factor
```

Block size increase factor for find.

Definition at line 151 of file settings.h.

#### 5.166.3.7 `find_initial_block_size`

```
\_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size
```

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

#### 5.166.3.8 find\_maximum\_block\_size

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::find\_maximum\_block\_size

Maximal block size for find.

Definition at line 157 of file settings.h.

#### 5.166.3.9 find\_scale\_factor

float \_\_gnu\_parallel::\_Settings::find\_scale\_factor

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by \_\_gnu\_parallel::\_find\_template().

#### 5.166.3.10 find\_sequential\_search\_size

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::find\_sequential\_search\_size

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by \_\_gnu\_parallel::\_find\_template().

#### 5.166.3.11 for\_each\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::for\_each\_minimal\_n

Minimal input size for for\_each.

Definition at line 163 of file settings.h.

#### 5.166.3.12 generate\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::generate\_minimal\_n

Minimal input size for generate.

Definition at line 166 of file settings.h.

**5.166.3.13 L1\_cache\_size**

```
unsigned long long __gnu_parallel::_Settings::L1_cache_size
```

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

**5.166.3.14 L2\_cache\_size**

```
unsigned long long __gnu_parallel::_Settings::L2_cache_size
```

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

**5.166.3.15 max\_element\_minimal\_n**

```
\_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n
```

Minimal input size for `max_element`.

Definition at line 169 of file settings.h.

**5.166.3.16 merge\_minimal\_n**

```
\_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n
```

Minimal input size for `merge`.

Definition at line 172 of file settings.h.

**5.166.3.17 merge\_oversampling**

```
unsigned int __gnu_parallel::_Settings::merge_oversampling
```

Oversampling factor for `merge`.

Definition at line 175 of file settings.h.

Referenced by `__gnu_parallel::multiway_merge_sampling_splitting()`.

#### 5.166.3.18 min\_element\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::min\_element\_minimal\_n

Minimal input size for min\_element.

Definition at line 178 of file settings.h.

#### 5.166.3.19 multiway\_merge\_minimal\_k

int \_\_gnu\_parallel::\_Settings::multiway\_merge\_minimal\_k

Oversampling factor for multiway\_merge.

Definition at line 184 of file settings.h.

#### 5.166.3.20 multiway\_merge\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::multiway\_merge\_minimal\_n

Minimal input size for multiway\_merge.

Definition at line 181 of file settings.h.

#### 5.166.3.21 multiway\_merge\_oversampling

unsigned int \_\_gnu\_parallel::\_Settings::multiway\_merge\_oversampling

Oversampling factor for multiway\_merge.

Definition at line 187 of file settings.h.

#### 5.166.3.22 nth\_element\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::nth\_element\_minimal\_n

Minimal input size for nth\_element.

Definition at line 190 of file settings.h.

Referenced by \_\_gnu\_parallel::\_parallel\_nth\_element().

**5.166.3.23 partial\_sort\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partial_sort_minimal_n`

Minimal input size for `partial_sort`.

Definition at line 203 of file `settings.h`.

**5.166.3.24 partial\_sum\_dilation**

`float __gnu_parallel::_Settings::partial_sum_dilation`

Ratio for `partial_sum`. Assume "sum and write result" to be this factor slower than just "sum".

Definition at line 207 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

**5.166.3.25 partial\_sum\_minimal\_n**

`unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n`

Minimal input size for `partial_sum`.

Definition at line 210 of file `settings.h`.

**5.166.3.26 partition\_chunk\_share**

`double __gnu_parallel::_Settings::partition_chunk_share`

Chunk size for partition, relative to input size. If  $> 0.0$ , this value overrides `partition_chunk_size`.

Definition at line 197 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_partition()`.

**5.166.3.27 partition\_chunk\_size**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partition_chunk_size`

Chunk size for partition.

Definition at line 193 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_partition()`.

#### 5.166.3.28 partition\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::partition\_minimal\_n

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by \_\_gnu\_parallel::\_parallel\_nth\_element().

#### 5.166.3.29 qsb\_steals

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::qsb\_steals

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

#### 5.166.3.30 random\_shuffle\_minimal\_n

unsigned int \_\_gnu\_parallel::\_Settings::random\_shuffle\_minimal\_n

Minimal input size for random\_shuffle.

Definition at line 213 of file settings.h.

#### 5.166.3.31 replace\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::replace\_minimal\_n

Minimal input size for replace and replace\_if.

Definition at line 216 of file settings.h.

#### 5.166.3.32 search\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::search\_minimal\_n

Minimal input size for search and search\_n.

Definition at line 273 of file settings.h.



**5.166.3.33 set\_difference\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Definition at line 219 of file `settings.h`.

**5.166.3.34 set\_intersection\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

**5.166.3.35 set\_symmetric\_difference\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

**5.166.3.36 set\_union\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

**5.166.3.37 sort\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

#### 5.166.3.38 sort\_mwms\_oversampling

```
unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling
```

Oversampling factor for parallel std::sort (MWMS).

Definition at line 234 of file settings.h.

Referenced by \_\_gnu\_parallel::parallel\_sort\_mwms\_pu().

#### 5.166.3.39 sort\_qs\_num\_samples\_preset

```
unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset
```

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file settings.h.

#### 5.166.3.40 sort\_qsb\_base\_case\_maximal\_n

```
\_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n
```

Maximal subsequence \_\_length to switch to unbalanced \_\_base case. Applies to std::sort with dynamically load-balanced quicksort.

Definition at line 241 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_qsb\_local\_sort\_with\_helping().

#### 5.166.3.41 TLB\_size

```
unsigned int __gnu_parallel::_Settings::TLB_size
```

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs(), and \_\_gnu\_parallel::\_\_sequential\_random\_shuffle().

#### 5.166.3.42 transform\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::transform\_minimal\_n

Minimal input size for parallel std::transform.

Definition at line 244 of file settings.h.

#### 5.166.3.43 unique\_copy\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::unique\_copy\_minimal\_n

Minimal input size for unique\_copy.

Definition at line 247 of file settings.h.

The documentation for this struct was generated from the following file:

- [settings.h](#)

### 5.167 \_\_gnu\_parallel::\_SplitConsistently< \_\_exact, \_RAIter, \_Compare, \_SortingPlacesIterator > Struct Template Reference

#### 5.167.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file multiway\_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

### 5.168 \_\_gnu\_parallel::\_SplitConsistently< false, \_RAIter, \_Compare, \_SortingPlacesIterator > Struct Template Reference

#### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWMSortingData](#)< \_RAIter > \* \_\_sd, \_Compare & \_\_comp, const typename std::iterator\_traits< \_RAIter >::difference\_type \_\_num\_samples) const

#### 5.168.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

#### 5.169 `__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

##### Public Member Functions

- `void operator() (const \_ThreadIndex __iam, \_PMWMSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const`

#### 5.169.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

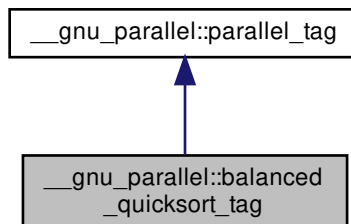
Definition at line 128 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

#### 5.170 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



## Public Member Functions

- **balanced\_quicksort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

### 5.170.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

### 5.170.2 Member Function Documentation

#### 5.170.2.1 [\\_\\_get\\_num\\_threads](#)()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)().

#### 5.170.2.2 [set\\_num\\_threads](#)()

```
void __gnu_parallel::parallel_tag::set_num_threads (  
    \_ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<a href="#">__num_threads</a>	Desired number of threads.
-------------------------------	----------------------------

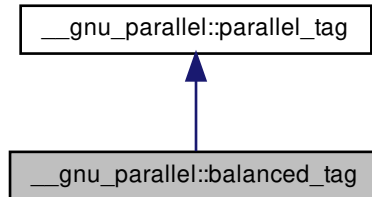
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.171 \_\_gnu\_parallel::balanced\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::balanced\_tag:



### Public Member Functions

- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) [\\_\\_num\\_threads](#))

#### 5.171.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file [tags.h](#).

#### 5.171.2 Member Function Documentation

##### 5.171.2.1 \_\_get\_num\_threads()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file [tags.h](#).

Referenced by `__gnu_parallel::__parallel_sort()`.

##### 5.171.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex \_\_num\_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

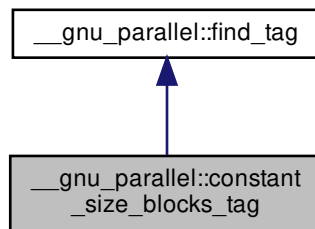
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.172 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



#### 5.172.1 Detailed Description

Selects the constant block size variant for `std::find()`.

#### See also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

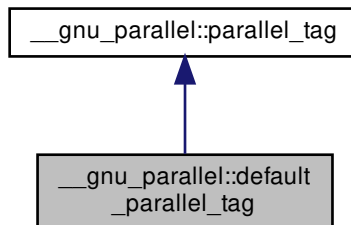
Definition at line 178 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.173 \_\_gnu\_parallel::default\_parallel\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::default\_parallel\_tag:



## Public Member Functions

- **default\_parallel\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

## 5.173.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file tags.h.

## 5.173.2 Member Function Documentation

## 5.173.2.1 \_\_get\_num\_threads()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

## 5.173.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.



#### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

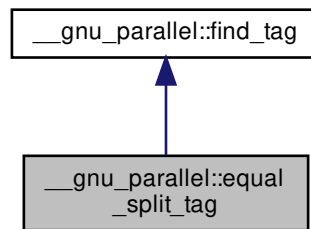
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.174 `__gnu_parallel::equal_split_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::equal_split_tag`:



#### 5.174.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

#### See also

`_GLIBCXX_FIND_EQUAL_SPLIT`

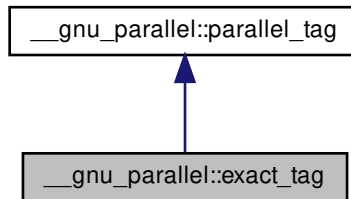
Definition at line 182 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

5.175 `__gnu_parallel::exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::exact_tag`:



## Public Member Functions

- `exact_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

## 5.175.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

## 5.175.2 Member Function Documentation

5.175.2.1 `__get_num_threads()`

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.175.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

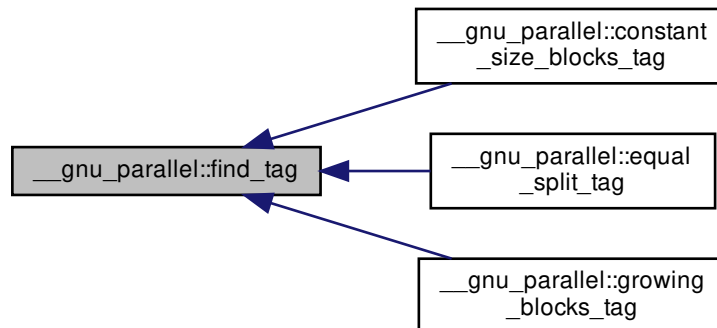
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.176 \_\_gnu\_parallel::find\_tag Struct Reference**

Inheritance diagram for `__gnu_parallel::find_tag`:

**5.176.1 Detailed Description**

Base class for for `std::find()` variants.

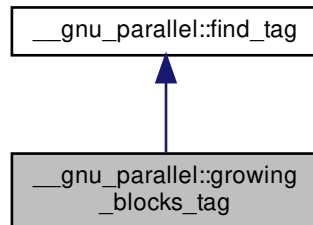
Definition at line 104 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.177 `__gnu_parallel::growing_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



#### 5.177.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_GROWING_BLOCKS`

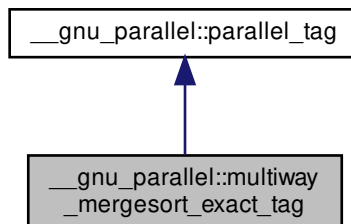
Definition at line 174 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.178 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



## Public Member Functions

- **multiway\_mergesort\_exact\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

### 5.178.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

### 5.178.2 Member Function Documentation

#### 5.178.2.1 [\\_\\_get\\_num\\_threads](#)()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)().

#### 5.178.2.2 [set\\_num\\_threads](#)()

```
void __gnu_parallel::parallel_tag::set_num_threads (  
    \_ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<a href="#">__num_threads</a>	Desired number of threads.
-------------------------------	----------------------------

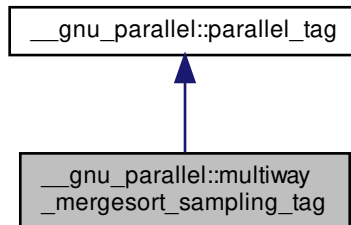
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.179 \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag:



### Public Member Functions

- **multiway\_mergesort\_sampling\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

#### 5.179.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

#### 5.179.2 Member Function Documentation

##### 5.179.2.1 \_\_get\_num\_threads()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.179.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

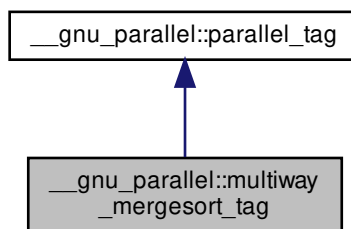
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.180 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



#### Public Member Functions

- **`multiway_mergesort_tag`** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

### 5.180.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file `tags.h`.

## 5.180.2 Member Function Documentation

### 5.180.2.1 \_\_get\_num\_threads()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_sort().

### 5.180.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<code><i>__num_threads</i></code>	Desired number of threads.
-----------------------------------	----------------------------

Definition at line 73 of file tags.h.

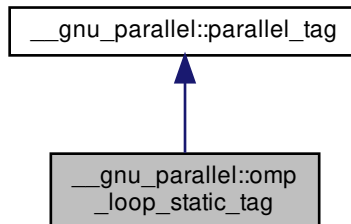
The documentation for this struct was generated from the following file:

- [tags.h](#)



## 5.181 \_\_gnu\_parallel::omp\_loop\_static\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_static\_tag:



### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads \( \)](#)
- void [set\\_num\\_threads \( \\_ThreadIndex \\_\\_num\\_threads \)](#)

#### 5.181.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

#### 5.181.2 Member Function Documentation

##### 5.181.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

##### 5.181.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

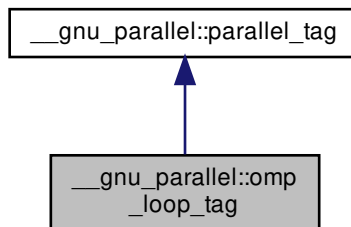
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.182 \_\_gnu\_parallel::omp\_loop\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_tag:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads \(\)](#)
- void [set\\_num\\_threads \(\\_ThreadIndex \\_\\_num\\_threads\)](#)

## 5.182.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

## 5.182.2 Member Function Documentation

#### 5.182.2.1 `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.182.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

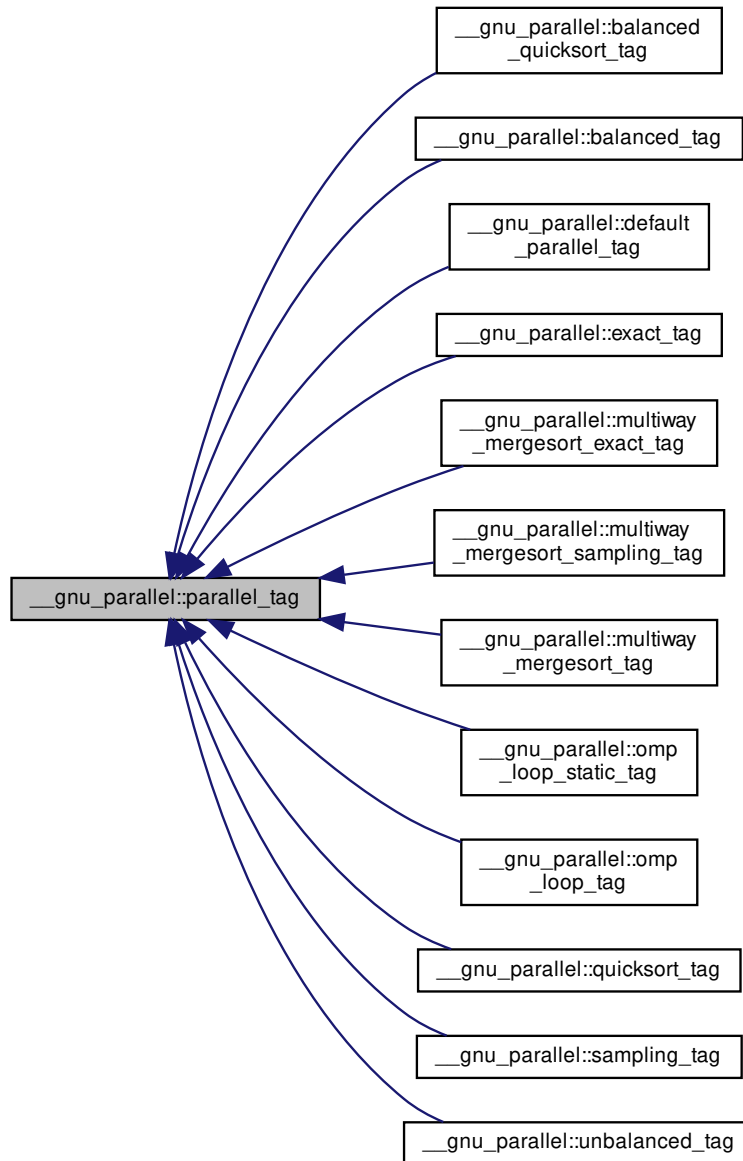
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.183 \_\_gnu\_parallel::parallel\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::parallel\_tag:

**Public Member Functions**

- [parallel\\_tag](#) ()
- [parallel\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

### 5.183.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

### 5.183.2 Constructor & Destructor Documentation

#### 5.183.2.1 `parallel_tag()` [1/2]

```
__gnu_parallel::parallel_tag::parallel_tag ( ) [inline]
```

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

#### 5.183.2.2 `parallel_tag()` [2/2]

```
__gnu_parallel::parallel_tag::parallel_tag (
    _ThreadIndex __num_threads ) [inline]
```

Default constructor. Recommend number of threads to use.

#### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

Definition at line 58 of file tags.h.

### 5.183.3 Member Function Documentation

#### 5.183.3.1 `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

## 5.183.3.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
    _ThreadIndex __num_threads ) [inline]
```

Set the desired number of threads.

## Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

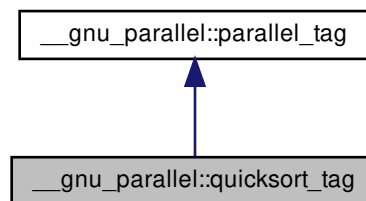
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.184 \_\_gnu\_parallel::quicksort\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::quicksort\_tag:



## Public Member Functions

- **quicksort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

## 5.184.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

## 5.184.2 Member Function Documentation

### 5.184.2.1 `__get_num_threads()`

`__ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]`

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.184.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    __ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

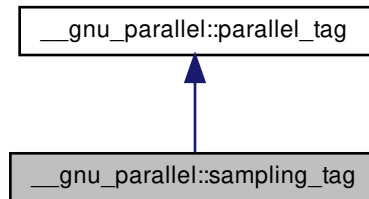
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

5.185 `__gnu_parallel::sampling_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::sampling_tag`:



## Public Member Functions

- **sampling\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

## 5.185.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

## 5.185.2 Member Function Documentation

5.185.2.1 `__get_num_threads()`

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.185.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex __num_threads ) [inline], [inherited]
```

Set the desired number of threads.



**Parameters**

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.186 \_\_gnu\_parallel::sequential\_tag Struct Reference****5.186.1 Detailed Description**

Forces sequential execution at compile time.

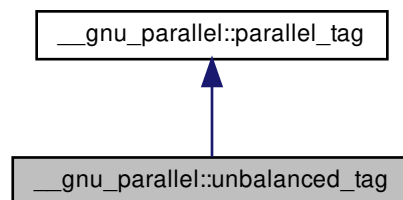
Definition at line 42 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

**5.187 \_\_gnu\_parallel::unbalanced\_tag Struct Reference**

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:

**Public Member Functions**

- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) `__num_threads`)

### 5.187.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

### 5.187.2 Member Function Documentation

#### 5.187.2.1 \_\_get\_num\_threads()

```
\_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.187.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
    \_ThreadIndex \_\_num\_threads ) [inline], [inherited]
```

Set the desired number of threads.

#### Parameters

<code><a href="#">__num_threads</a></code>	Desired number of threads.
--	----------------------------

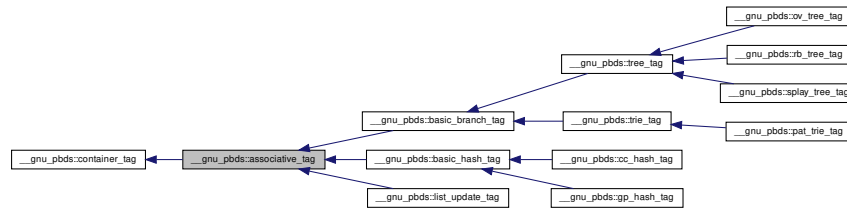
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.188 \_\_gnu\_pbds::associative\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::associative\_tag:



### 5.188.1 Detailed Description

Basic associative-container.

Definition at line 135 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.189 \_\_gnu\_pbds::basic\_branch< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc > Class Template Reference

Inherits type< Key, Mapped, \_Alloc, Tag, Policy\_Tl >.

### Public Types

- typedef Node\_Update **node\_update**

### Protected Member Functions

- **basic\_branch** (const [basic\\_branch](#) &other)
- template<typename T0 >  
**basic\_branch** (T0 t0)
- template<typename T0 , typename T1 >  
**basic\_branch** (T0 t0, T1 t1)
- template<typename T0 , typename T1 , typename T2 >  
**basic\_branch** (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 , typename T2 , typename T3 >  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

## 5.189.1 Detailed Description

```
template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_TI, typename _Alloc>
class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_TI, _Alloc >
```

A branched, tree-like (tree, trie) container abstraction.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Tag</i>	Instantiating data structure type, see container_tag.
<i>Node_Update</i>	Updates nodes, restores invariants.
<i>Policy_TL</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: tree\_tag, trie\_tag, and their descendants.

Base choices are: detail::ov\_tree\_map, detail::rb\_tree\_map, detail::splay\_tree\_map, and detail::pat\_trie\_map.

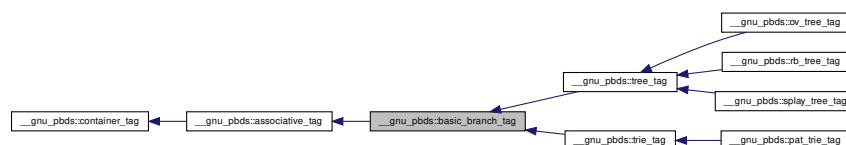
Definition at line 555 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.190 \_\_gnu\_pbds::basic\_branch\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::basic\_branch\_tag:



## 5.190.1 Detailed Description

Basic branch structure.

Definition at line 147 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.191 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >` Class Template Reference

Inherits type< Key, Mapped, \_Alloc, Tag, \_\_gnu\_cxx::typelist::append< \_\_gnu\_cxx::typelist::create4< Hash\_Fn, Eq\_Fn, Resize\_Policy, detail::integral\_constant< int, Store\_Hash > >::type, Policy\_Tl >::type >.

### Protected Member Functions

- **basic\_hash\_table** (const [basic\\_hash\\_table](#) &other)
- template<typename T0 >  
**basic\_hash\_table** (T0 t0)
- template<typename T0 , typename T1 >  
**basic\_hash\_table** (T0 t0, T1 t1)
- template<typename T0 , typename T1 , typename T2 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 , typename T2 , typename T3 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

### 5.191.1 Detailed Description

template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename Resize\_Policy, bool Store\_Hash, typename Tag, typename Policy\_Tl, typename \_Alloc>

class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`

A hashed container abstraction.

#### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Policy_Tl</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: cc\_hash\_tag, gp\_hash\_tag, and descendants of basic\_hash\_tag.

Base choices are: detail::cc\_ht\_map, detail::gp\_ht\_map

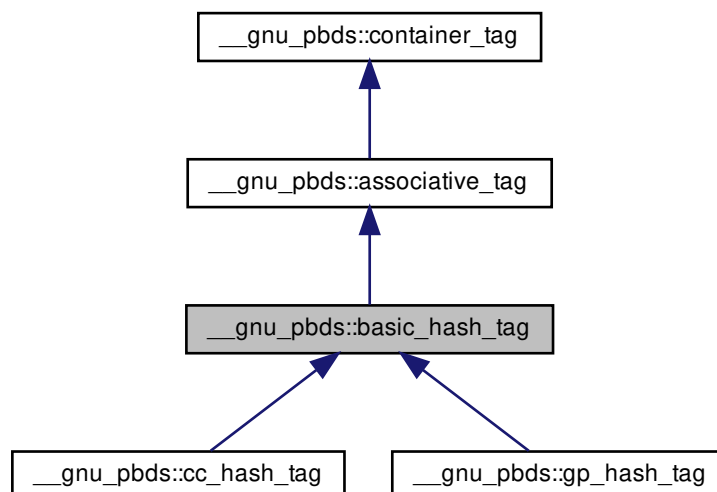
Definition at line 104 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.192 \_\_gnu\_pbds::basic\_hash\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::basic\_hash\_tag:



### 5.192.1 Detailed Description

Basic hash structure.

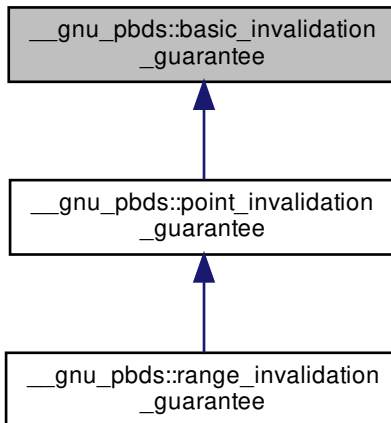
Definition at line 138 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.193 `__gnu_pbds::basic_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



#### 5.193.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

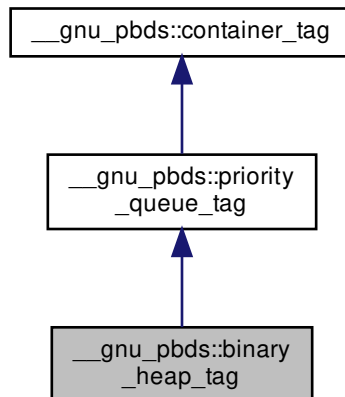
Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.194 `__gnu_pbds::binary_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



## 5.194.1 Detailed Description

Binary-heap (array-based).

Definition at line 183 of file `tag_and_trait.hpp`.

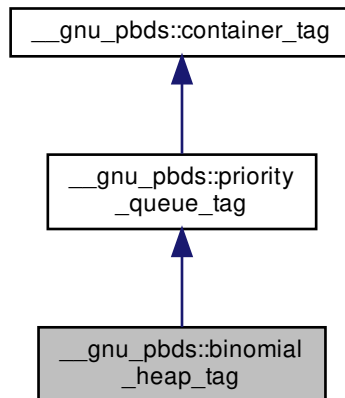
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



### 5.195 `__gnu_pbds::binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



#### 5.195.1 Detailed Description

Binomial-heap.

Definition at line 177 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.196 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

#### Public Types

- enum { [external\\_load\\_access](#) }
- typedef `Size_Type` **size\_type**

#### Public Member Functions

- [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#) (float load=0.5)
- float [get\\_load](#) () const
- void [set\\_load](#) (float load)
- void **swap** ([cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)< `External_Load_Access`, `Size_Type` > &other)

## Protected Member Functions

- bool `is_grow_needed` (size\_type size, size\_type num\_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (size\_type num\_entries)
- void `notify_externally_resized` (size\_type new\_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size\_type num\_entries)
- void `notify_resized` (size\_type new\_size)

### 5.196.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

Definition at line 293 of file hash\_policy.hpp.

### 5.196.2 Member Enumeration Documentation

#### 5.196.2.1 anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

#### Enumerator

<code>external_load_access</code>	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.
-----------------------------------	---

Definition at line 298 of file hash\_policy.hpp.

### 5.196.3 Constructor & Destructor Documentation

#### 5.196.3.1 cc\_hash\_max\_collision\_check\_resize\_trigger()

```
template<bool External_Load_Access, typename Size_Type >
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_hash_max_collision_
(
    float load = 0.5 )
```

Default constructor, or constructor taking load, a \_\_load factor which it will attempt to maintain.

Definition at line 44 of file hash\_policy.hpp.

### 5.196.4 Member Function Documentation

#### 5.196.4.1 get\_load()

```
template<bool External_Load_Access, typename Size_Type >
float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↔
::get_load ( ) const [inline]
```

Returns the current load.

Definition at line 190 of file hash\_policy.hpp.

#### 5.196.4.2 is\_grow\_needed()

```
template<bool External_Load_Access, typename Size_Type >
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↔
::is_grow_needed (
    size_type size,
    size_type num_entries ) const [inline], [protected]
```

Queries whether a grow is needed. This method is called only if this object indicated is needed.

Definition at line 133 of file hash\_policy.hpp.

#### 5.196.4.3 is\_resize\_needed()

```
template<bool External_Load_Access, typename Size_Type >  
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵  
::is_resize_needed ( ) const [inline], [protected]
```

Queries whether a resize is needed.

Definition at line 127 of file hash\_policy.hpp.

#### 5.196.4.4 notify\_cleared()

```
template<bool External_Load_Access, typename Size_Type >  
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵  
::notify_cleared ( ) [protected]
```

Notifies the table was cleared.

Definition at line 121 of file hash\_policy.hpp.

#### 5.196.4.5 notify\_erase\_search\_collision()

```
template<bool External_Load_Access, typename Size_Type >  
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵  
::notify_erase_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

Definition at line 97 of file hash\_policy.hpp.

#### 5.196.4.6 notify\_erase\_search\_end()

```
template<bool External_Load_Access, typename Size_Type >  
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵  
::notify_erase_search_end ( ) [inline], [protected]
```

Notifies a search ended.

Definition at line 103 of file hash\_policy.hpp.

#### 5.196.4.7 notify\_erase\_search\_start()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_start ( ) [inline], [protected]
```

Notifies an erase search started.

Definition at line 91 of file hash\_policy.hpp.

#### 5.196.4.8 notify\_erased()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erased (
    size_type num_entries ) [inline], [protected]
```

Notifies an element was erased.

Definition at line 115 of file hash\_policy.hpp.

#### 5.196.4.9 notify\_externally\_resized()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_externally_resized (
    size_type new_size ) [protected]
```

Notifies the table was resized externally.

Definition at line 172 of file hash\_policy.hpp.

#### 5.196.4.10 notify\_find\_search\_collision()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

Definition at line 61 of file hash\_policy.hpp.

#### 5.196.4.11 `notify_find_search_end()`

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_end ( ) [inline], [protected]
```

Notifies a search ended.

Definition at line 67 of file `hash_policy.hpp`.

#### 5.196.4.12 `notify_find_search_start()`

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_start ( ) [inline], [protected]
```

Notifies a find search started.

Definition at line 55 of file `hash_policy.hpp`.

#### 5.196.4.13 `notify_insert_search_collision()`

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

Definition at line 79 of file `hash_policy.hpp`.

#### 5.196.4.14 `notify_insert_search_end()`

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_end ( ) [inline], [protected]
```

Notifies a search ended.

Definition at line 85 of file `hash_policy.hpp`.

#### 5.196.4.15 notify\_insert\_search\_start()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_start ( ) [inline], [protected]
```

Notifies an insert search started.

Definition at line 73 of file hash\_policy.hpp.

#### 5.196.4.16 notify\_inserted()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_inserted (
    size_type num_entries ) [inline], [protected]
```

Notifies an element was inserted.

Definition at line 109 of file hash\_policy.hpp.

#### 5.196.4.17 notify\_resized()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_resized (
    size_type new_size ) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 139 of file hash\_policy.hpp.

#### 5.196.4.18 set\_load()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::set_load (
    float load )
```

Sets the load; does not resize the container.

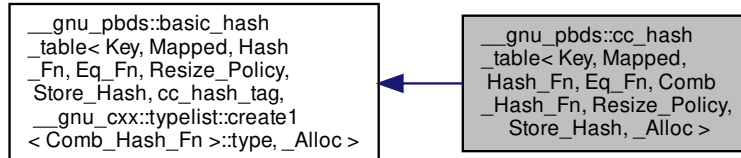
Definition at line 205 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

5.197 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn**
- typedef `cc_hash_tag` **container\_category**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `Resize_Policy` **resize\_policy**

#### Public Member Functions

- `cc_hash_table` ()
- `cc_hash_table` (const hash\_fn &h)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It >  
  `cc_hash_table` (It first, It last)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- **cc\_hash\_table** (const `cc_hash_table` &other)
- `cc_hash_table` & **operator=** (const `cc_hash_table` &other)
- void **swap** (`cc_hash_table` &other)



### 5.197.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_↵
_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename
_Alloc = std::allocator<char>>>
class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A collision-chaining hash-based associative container.

#### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Hash_Fn</i>	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
<i>_Alloc</i>	Allocator type.

Base tag choices are: cc\_hash\_tag.

Base is basic\_hash\_table.

Definition at line 204 of file assoc\_container.hpp.

### 5.197.2 Constructor & Destructor Documentation

#### 5.197.2.1 cc\_hash\_table() [1/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table ( ) [inline]
```

Default constructor.

Definition at line 217 of file assoc\_container.hpp.

5.197.2.2 `cc_hash_table()` [2/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `Hash_Fn` object of the container object.

Definition at line 221 of file `assoc_container.hpp`.

5.197.2.3 `cc_hash_table()` [3/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h,
    const eq_fn & e ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 228 of file `assoc_container.hpp`.

5.197.2.4 `cc_hash_table()` [4/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 236 of file `assoc_container.hpp`.

5.197.2.5 `cc_hash_table()` [5/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch,
    const resize_policy & rp ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 245 of file `assoc_container.hpp`.

5.197.2.6 `cc_hash_table()` [6/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last ) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 253 of file `assoc_container.hpp`.

5.197.2.7 `cc_hash_table()` [7/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
```

```

    It last,
    const hash_fn & h ) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 260 of file `assoc_container.hpp`.

#### 5.197.2.8 `cc_hash_table()` [8/10]

```

template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e ) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 271 of file `assoc_container.hpp`.

#### 5.197.2.9 `cc_hash_table()` [9/10]

```

template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch ) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 283 of file `assoc_container.hpp`.

### 5.197.2.10 cc\_hash\_table() [10/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_hash_fn & ch,
    const resize_policy & rp ) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object, and r\_resize\_policy will be copied by the resize\_policy object of the container object.

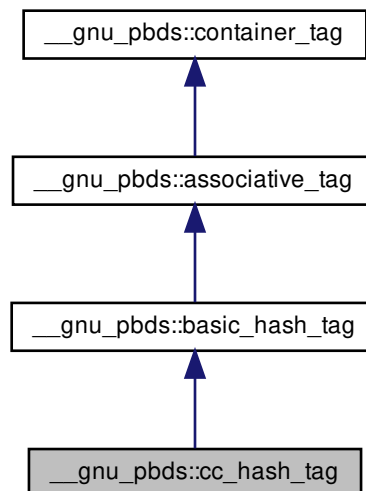
Definition at line 297 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.198 \_\_gnu\_pbds::cc\_hash\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::cc\_hash\_tag:



## 5.198.1 Detailed Description

Collision-chaining hash.

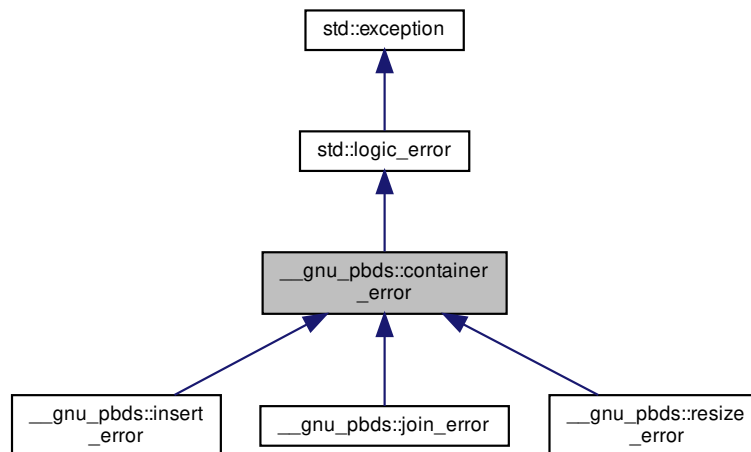
Definition at line 141 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.199 \_\_gnu\_pbds::container\_error Struct Reference

Inheritance diagram for \_\_gnu\_pbds::container\_error:



## Public Member Functions

- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.199.1 Detailed Description

Base class for exceptions.

Definition at line 57 of file exception.hpp.

## 5.199.2 Member Function Documentation

### 5.199.2.1 what()

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

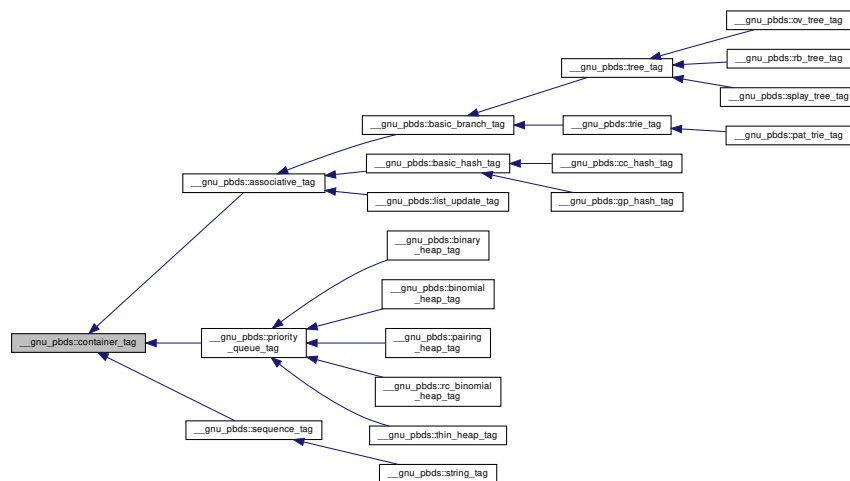
Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

## 5.200 \_\_gnu\_pbds::container\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::container\_tag:



### 5.200.1 Detailed Description

Base data structure tag.

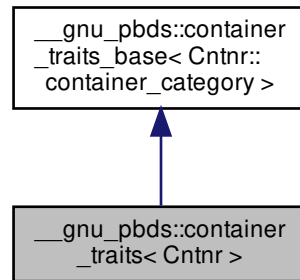
Definition at line 125 of file [tag\\_and\\_trait.hpp](#).

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.201 \_\_gnu\_pbds::container\_traits&lt; Cntnr &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::container\_traits< Cntnr >:



## Public Types

- enum { [order\\_preserving](#), [erase\\_can\\_throw](#), [split\\_join\\_can\\_throw](#), [reverse\\_iteration](#) }
- typedef [container\\_traits\\_base](#)< container\_category > **base\_type**
- typedef Cntnr::container\_category **container\_category**
- typedef Cntnr **container\_type**
- typedef base\_type::invalidation\_guarantee **invalidation\_guarantee**

## 5.201.1 Detailed Description

```
template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

Definition at line 418 of file tag\_and\_trait.hpp.

## 5.201.2 Member Enumeration Documentation

## 5.201.2.1 anonymous enum

```
template<typename Cntnr >
anonymous enum
```



### Enumerator

order_preserving	True only if Cntnr objects guarantee storing keys by order.
erase_can_throw	True only if erasing a key can throw.
split_join_can_throw	True only if split or join operations can throw.
reverse_iteration	True only reverse iterators are supported.

Definition at line 426 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.202 \_\_gnu\_pbds::container\_traits\_base< \_Tag > Struct Template Reference

### 5.202.1 Detailed Description

```
template<typename _Tag>
struct __gnu_pbds::container_traits_base< _Tag >
```

Primary template, container traits base.

Definition at line 220 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.203 \_\_gnu\_pbds::container\_traits\_base< binary\_heap\_tag > Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [binary\\_heap\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 5.203.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< binary_heap_tag >
```

Specialization, binary heap.

Definition at line 400 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.204 `__gnu_pbds::container_traits_base< binomial_heap_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [binomial\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 5.204.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< binomial_heap_tag >
```

Specialization, binomial heap.

Definition at line 368 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.205 `__gnu_pbds::container_traits_base< cc_hash_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 5.205.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< cc_hash_tag >
```

Specialization, cc hash.

Definition at line 224 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.206 `__gnu_pbds::container_traits_base< gp_hash_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 5.206.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< gp_hash_tag >
```

Specialization, gp hash.

Definition at line 240 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.207 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [list\\_update\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 5.207.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< list_update_tag >
```

Specialization, list update.

Definition at line 320 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.208 `__gnu_pbds::container_traits_base< ov_tree_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [ov\\_tree\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 5.208.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< ov_tree_tag >
```

Specialization, ov tree.

Definition at line 288 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.209 `__gnu_pbds::container_traits_base< pairing_heap_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [pairing\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 5.209.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< pairing_heap_tag >
```

Specialization, pairing heap.

Definition at line 336 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.210 `__gnu_pbds::container_traits_base< pat_trie_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [pat\\_trie\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 5.210.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< pat_trie_tag >
```

Specialization, pat trie.

Definition at line 304 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.211 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [rb\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 5.211.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< rb_tree_tag >
```

Specialization, rb tree.

Definition at line 256 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.212 `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [rc\\_binomial\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 5.212.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
```

Specialization, rc binomial heap.

Definition at line 384 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.213 `__gnu_pbds::container_traits_base< splay_tree_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [splay\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 5.213.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< splay_tree_tag >
```

Specialization, splay tree.

Definition at line 272 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.214 `__gnu_pbds::container_traits_base< thin_heap_tag >` Struct Template Reference

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef `thin_heap_tag` **container\_category**
- typedef `point_invalidation_guarantee` **invalidation\_guarantee**

### 5.214.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< thin_heap_tag >
```

Specialization, thin heap.

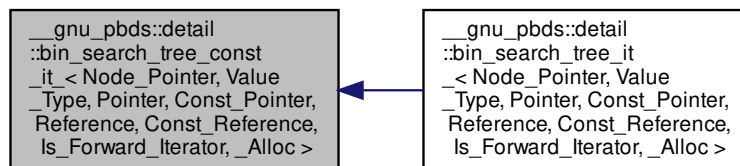
Definition at line 352 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.215 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



### Public Types

- typedef `Const_Pointer` **const\_pointer**
- typedef `Const_Reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::bidirectional_iterator_tag` **iterator\_category**
- typedef `Pointer` **pointer**
- typedef `Reference` **reference**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `bin_search_tree_const_it_` (const Node\_Pointer p\_nd=0)
- `bin_search_tree_const_it_` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other)
- `bool operator!=` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other) const
- `bool operator!=` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other) const
- `const_reference operator*` () const
- `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` & `operator++` ()
- `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` `operator++` (int)
- `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` & `operator--` ()
- `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` `operator--` (int)
- `const_pointer operator->` () const
- `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` & `operator=` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other)
- `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` & `operator=` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other)
- `bool operator==` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other) const
- `bool operator==` (const `bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` &other) const

## Public Attributes

- Node\_Pointer `m_p_nd`

## Protected Member Functions

- `void dec` (false\_type)
- `void dec` (true\_type)
- `void inc` (false\_type)
- `void inc` (true\_type)

### 5.215.1 Detailed Description

```
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_
Reference, Is_Forward_Iterator, _Alloc >
```

Const iterator.

Definition at line 105 of file `point_iterators.hpp`.

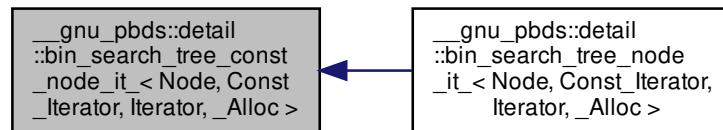
The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)



## 5.216 `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



### Public Types

- typedef `Const_Iterator` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` `metadata_const_reference`
- typedef `Node::metadata_type` `metadata_type`
- typedef `Const_Iterator` `reference`
- typedef `Const_Iterator` `value_type`

### Public Member Functions

- `bin_search_tree_const_node_it_` (`const node_pointer p_nd=0`)
- `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` `get_l_child` () const
- `metadata_const_reference` `get_metadata` () const
- `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` `get_r_child` () const
- `bool operator!=` (`const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other`) const
- `const_reference operator*` () const
- `bool operator==` (`const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other`) const

### Public Attributes

- `node_pointer` `m_p_nd`

#### 5.216.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Const node iterator.

Definition at line 58 of file `bin_search_tree_/node_iterators.hpp`.

## 5.216.2 Member Typedef Documentation

### 5.216.2.1 `const_reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >  
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,  
Iterator, _Alloc >::const_reference
```

Iterator's `__const` reference type.

Definition at line 80 of file `bin_search_tree_/node_iterators.hpp`.

### 5.216.2.2 `difference_type`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >  
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<  
Node, Const_Iterator, Iterator, _Alloc >::difference_type
```

Difference type.

Definition at line 71 of file `bin_search_tree_/node_iterators.hpp`.

### 5.216.2.3 `iterator_category`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >  
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,  
Iterator, Iterator, _Alloc >::iterator_category
```

Category.

Definition at line 68 of file `bin_search_tree_/node_iterators.hpp`.

### 5.216.2.4 `metadata_const_reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >  
typedef _Alloc::template rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<  
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

Definition at line 88 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.216.2.5 metadata\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Node::metadata_type \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_↵
Iterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

Definition at line 83 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.216.2.6 reference

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Const_Iterator \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator,
Iterator, _Alloc >::reference
```

Iterator's reference type.

Definition at line 77 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.216.2.7 value\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Const_Iterator \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const_Iterator,
Iterator, _Alloc >::value_type
```

Iterator's value type.

Definition at line 74 of file `bin_search_tree_/node_iterators.hpp`.

### 5.216.3 Member Function Documentation

#### 5.216.3.1 get\_l\_child()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin\_search\_tree\_const\_node\_it\_<Node, Const_Iterator, Iterator, _Alloc> \_\_gnu\_pbds::detail::bin\_search\_tree\_const
Node, Const_Iterator, Iterator, _Alloc >::get_l_child ( ) const [inline]
```

Returns the `__const` node iterator associated with the left node.

Definition at line 107 of file `bin_search_tree_/node_iterators.hpp`.

### 5.216.3.2 get\_metadata()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata ( ) const [inline]
```

Metadata access.

Definition at line 102 of file bin\_search\_tree\_/node\_iterators.hpp.

### 5.216.3.3 get\_r\_child()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_const_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_
Node, Const_Iterator, Iterator, _Alloc >::get_r_child ( ) const [inline]
```

Returns the \_\_const node iterator associated with the right node.

Definition at line 112 of file bin\_search\_tree\_/node\_iterators.hpp.

### 5.216.3.4 operator!=(())

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
    const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
    other ) const [inline]
```

Compares (negatively) to a different iterator object.

Definition at line 122 of file bin\_search\_tree\_/node\_iterators.hpp.

### 5.216.3.5 operator\*()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::operator* ( ) const [inline]
```

Access.

Definition at line 97 of file bin\_search\_tree\_/node\_iterators.hpp.

## 5.216.3.6 operator==()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
    const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
    other ) const [inline]
```

Compares to a different iterator object.

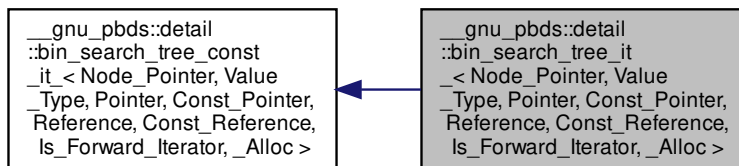
Definition at line 117 of file bin\_search\_tree\_/node\_iterators.hpp.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

## 5.217 \_\_gnu\_pbds::detail::bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >:



### Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

## Public Member Functions

- `bin_search_tree_it` (const Node\_Pointer p\_nd=0)
- `bin_search_tree_it` (const [bin\\_search\\_tree\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- `bool operator!=` (const [bin\\_search\\_tree\\_const\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- `bool operator!=` (const [bin\\_search\\_tree\\_const\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- `bin_search_tree_const_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > ::reference `operator*` () const
- `bin_search_tree_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & `operator++` ()
- `bin_search_tree_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > `operator++` (int)
- `bin_search_tree_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & `operator--` ()
- `bin_search_tree_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > `operator--` (int)
- `bin_search_tree_const_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > ::pointer `operator->` () const
- `bin_search_tree_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & `operator=` (const [bin\\_search\\_tree\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- `bin_search_tree_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & `operator=` (const [bin\\_search\\_tree\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- `bool operator==` (const [bin\\_search\\_tree\\_const\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- `bool operator==` (const [bin\\_search\\_tree\\_const\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const

## Public Attributes

- Node\_Pointer `m_p_nd`

## Protected Types

- typedef [bin\\_search\\_tree\\_const\\_it](#) < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > `base_it_type`

## Protected Member Functions

- void `dec` (false\_type)
- void `dec` (true\_type)
- void `inc` (false\_type)
- void `inc` (true\_type)

### 5.217.1 Detailed Description

```
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename
Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_
_Forward_Iterator, _Alloc >
```

Iterator.

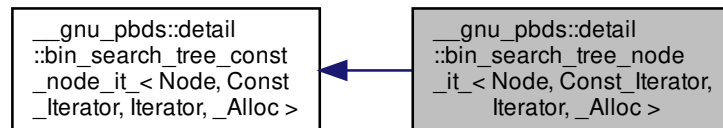
Definition at line 282 of file point\_iterators.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

### 5.218 \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >:



#### Public Types

- typedef Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) difference\_type
- typedef [trivial\\_iterator\\_tag](#) iterator\_category
- typedef \_Alloc::template rebind< [metadata\\_type](#) >::other::const\_reference [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value\\_type](#)

#### Public Member Functions

- **bin\_search\_tree\_node\_it\_** (const node\_pointer p\_nd=0)
- [bin\\_search\\_tree\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- [bin\\_search\\_tree\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > [get\\_r\\_child](#) () const
- bool [operator!=](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > &other) const
- Iterator [operator\\*](#) () const
- bool [operator==](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > &other) const

## Public Attributes

- node\_pointer `m_p_nd`

### 5.218.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 136 of file `bin_search_tree_/node_iterators.hpp`.

### 5.218.2 Member Typedef Documentation

#### 5.218.2.1 `const_reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::const_reference
```

Iterator's `__const` reference type.

Definition at line 153 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.218.2.2 `difference_type`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type [inherited]
```

Difference type.

Definition at line 71 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.218.2.3 `iterator_category`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::iterator_category [inherited]
```

Category.

Definition at line 68 of file `bin_search_tree_/node_iterators.hpp`.



#### 5.218.2.4 metadata\_const\_reference

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef _Alloc::template rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]
```

Const metadata reference type.

Definition at line 88 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 5.218.2.5 metadata\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::metadata_type [inherited]
```

Metadata type.

Definition at line 83 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 5.218.2.6 reference

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference
```

Iterator's reference type.

Definition at line 150 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 5.218.2.7 value\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type
```

Iterator's value type.

Definition at line 147 of file bin\_search\_tree\_/node\_iterators.hpp.

### 5.218.3 Member Function Documentation

#### 5.218.3.1 `get_l_child()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_l_child ( ) const [inline]
```

Returns the node iterator associated with the left node.

Definition at line 167 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.218.3.2 `get_metadata()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata ( ) const [inline], [inherited]
```

Metadata access.

Definition at line 102 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.218.3.3 `get_r_child()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child ( ) const [inline]
```

Returns the node iterator associated with the right node.

Definition at line 175 of file `bin_search_tree_/node_iterators.hpp`.

#### 5.218.3.4 `operator!=( )`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!=(
    const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
    other ) const [inline], [inherited]
```

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

5.218.3.5 `operator*()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >↵
::operator* ( ) const [inline]
```

Access.

Definition at line 162 of file `bin_search_tree_/node_iterators.hpp`.

5.218.3.6 `operator==()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc ↵
>::operator== (
    const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
    other ) const [inline], [inherited]
```

Compares to a different iterator object.

Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

5.219 `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`  
Struct Template Reference

## Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_↵ traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_↵ _update_pointer`
- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_↵ traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_↵ _traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits_↵ ::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_↵ _traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits_↵ ::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 5.219.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename
_Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
```

Binary search tree traits, primary template.

Definition at line 63 of file `bin_search_tree_/traits.hpp`.

### 5.219.2 Member Typedef Documentation

#### 5.219.2.1 `node_const_iterator`

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_↵
Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin\_search\_tree\_const\_node\_it< Node, point\_const\_iterator, point\_iterator, _Alloc> \_\_gnu\_pbds::detail::
Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node\_const\_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

## 5.220 `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

### Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#)< typename \_Alloc::template rebind< node >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_↵ traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **const\_reverse\_iterator**
- typedef Node **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#)< Node, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > [node\\_const\\_iterator](#)
- typedef [node\\_const\\_iterator](#) **node\_iterator**
- typedef Node\_Update< [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#)< [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node\_↵ \_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#)< typename \_Alloc::template rebind< node >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_↵ traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef [point\\_const\\_iterator](#) **point\_iterator**
- typedef [const\\_reverse\\_iterator](#) **reverse\_iterator**

### 5.220.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_CIttr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class
Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
```

Specialization.

Definition at line 169 of file `bin_search_tree_/traits.hpp`.

### 5.220.2 Member Typedef Documentation

#### 5.220.2.1 node\_const\_iterator

```
template<typename Key , class Cmp_Fn , template< typename Node_CIttr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node , typename _Alloc >
typedef bin_search_tree_const_node_it< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::node_const_iterator
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

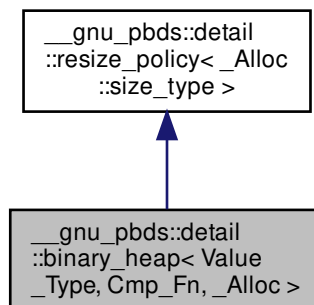
Definition at line 216 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

### 5.221 \_\_gnu\_pbds::detail::binary\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `cond_dealtor< value_type, _Alloc >` **cond\_dealtor\_t**
- typedef `binary_heap_const_iterator< value_type, entry, simple_value, _Alloc >` **const\_iterator**
- typedef `value_allocator::const_pointer` **const\_pointer**
- typedef `value_allocator::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `__conditional_type< simple_value, value_type, pointer >::__type` **entry**
- typedef `_Alloc::template rebind< entry >::other` **entry\_allocator**
- typedef `entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type` **entry\_cmp**
- typedef `entry_allocator::pointer` **entry\_pointer**
- typedef `const_iterator` **iterator**
- typedef `binary_heap_point_const_iterator< value_type, entry, simple_value, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `value_allocator::pointer` **pointer**
- typedef `value_allocator::reference` **reference**
- typedef `__gnu_pbds::detail::resize_policy< typename _Alloc::size_type >` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binary\_heap** (const `cmp_fn` &)
- **binary\_heap** (const `binary_heap` &)
- **iterator begin** ()
- **const\_iterator begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator end** ()
- **const\_iterator end** () const
- void **erase** (`point_iterator`)
- void **erase\_at** (`entry_pointer`, `size_type`, `false_type`)
- void **erase\_at** (`entry_pointer`, `size_type`, `true_type`)
- template<typename `Pred` >  
  `size_type` **erase\_if** (`Pred`)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- `size_type` **get\_new\_size\_for\_arbitrary** (`size_type`) const
- `size_type` **get\_new\_size\_for\_grow** () const
- `size_type` **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (`size_type`) const
- void **join** (`binary_heap` &)
- `size_type` **max\_size** () const
- void **modify** (`point_iterator`, `const_reference`)
- void **notify\_arbitrary** (`size_type`)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- void **pop** ()

- [point\\_iterator](#) **push** (const\_reference)
- bool **resize\_needed\_for\_grow** (size\_type) const
- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, [binary\\_heap](#) &)
- void **swap** ([resize\\_policy](#)<\_Alloc::size\_type > &)
- void **swap** ([binary\\_heap](#) &)
- const\_reference **top** () const

#### Static Public Attributes

- static const \_Alloc::size\_type **min\_size**

#### Protected Member Functions

- template<typename It >  
void **copy\_from\_range** (It, It)

#### 5.221.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binary heaps composed of resize and compare policies.

Based on CLRS.

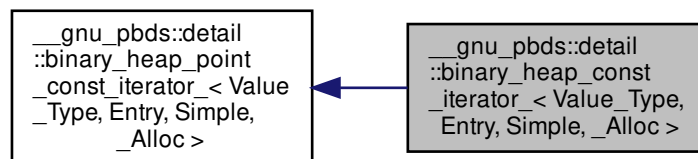
Definition at line 84 of file [binary\\_heap.hpp](#).

The documentation for this class was generated from the following file:

- [binary\\_heap.hpp](#)

#### 5.222 \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >:



### Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

### Public Member Functions

- `binary_heap_const_iterator_` (`entry_pointer p_e`)
- `binary_heap_const_iterator_` ()
- `binary_heap_const_iterator_` (`const binary_heap_const_iterator_ &other`)
- `bool operator!=` (`const binary_heap_const_iterator_ &other`) `const`
- `bool operator!=` (`const binary_heap_point_const_iterator_ &other`) `const`
- `const_reference operator*` () `const`
- `binary_heap_const_iterator_ & operator++` ()
- `binary_heap_const_iterator_ operator++` (`int`)
- `const_pointer operator->` () `const`
- `bool operator==` (`const binary_heap_const_iterator_ &other`) `const`
- `bool operator==` (`const binary_heap_point_const_iterator_ &other`) `const`

### Public Attributes

- `entry_pointer m_p_e`

#### 5.222.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_ < Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

Definition at line 60 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2 Member Typedef Documentation



#### 5.222.2.1 `const_pointer`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base\_type::const\_pointer \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value_Type,
Entry, Simple, _Alloc >::const\_pointer
```

Iterator's const pointer type.

Definition at line 80 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2.2 `const_reference`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base\_type::const\_reference \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value_Type,
Entry, Simple, _Alloc >::const\_reference
```

Iterator's const reference type.

Definition at line 86 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2.3 `difference_type`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef \_Alloc::difference\_type \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value_Type,
Entry, Simple, _Alloc >::difference\_type
```

Difference type.

Definition at line 71 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2.4 `iterator_category`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef std::forward\_iterator\_tag \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value_Type,
Entry, Simple, _Alloc >::iterator\_category
```

Category.

Definition at line 68 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2.5 pointer

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base\_type::pointer \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry,
Simple, \_Alloc >::pointer
```

Iterator's pointer type.

Definition at line 77 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2.6 reference

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base\_type::reference \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry,
Simple, \_Alloc >::reference
```

Iterator's reference type.

Definition at line 83 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.2.7 value\_type

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base\_type::value\_type \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry,
Simple, \_Alloc >::value\_type
```

Iterator's value type.

Definition at line 74 of file `binary_heap_/const_iterator.hpp`.

### 5.222.3 Constructor & Destructor Documentation

#### 5.222.3.1 `binary_heap_const_iterator_()` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
\_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >::binary\_heap\_const\_iterator\_
( ) [inline]
```

Default constructor.

Definition at line 94 of file `binary_heap_/const_iterator.hpp`.

### 5.222.3.2 `binary_heap_const_iterator_()` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_
(
    const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other )
[inline]
```

Copy constructor.

Definition at line 99 of file `binary_heap_/const_iterator.hpp`.

## 5.222.4 Member Function Documentation

### 5.222.4.1 `operator!=(())` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=
(
    const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other )
const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 110 of file `binary_heap_/const_iterator.hpp`.

### 5.222.4.2 `operator!=(())` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator!= (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

### 5.222.4.3 `operator*()`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* ( ) const [inline], [inherited]
```

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.222.4.4 `operator->()`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> ( ) const [inline], [inherited]
```

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.222.4.5 `operator==()` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==
(
    const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other )
const [inline]
```

Compares content to a different iterator object.

Definition at line 105 of file `binary_heap_/const_iterator.hpp`.

#### 5.222.4.6 `operator==()` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator== (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline], [inherited]
```

Compares content to a different iterator object.

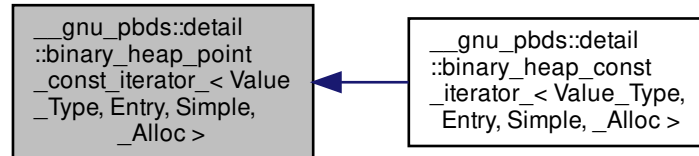
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/const\\_iterator.hpp](#)

## 5.223 `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



### Public Types

- `typedef _Alloc::template rebind< value_type >::other::const_pointer const_pointer`
- `typedef _Alloc::template rebind< value_type >::other::const_reference const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef _Alloc::template rebind< value_type >::other::pointer pointer`
- `typedef _Alloc::template rebind< value_type >::other::reference reference`
- `typedef Value_Type value_type`

### Public Member Functions

- `binary_heap_point_const_iterator_ (entry_pointer p_e)`
- `binary_heap_point_const_iterator_ ()`
- `binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_ &other)`
- `bool operator!= (const binary_heap_point_const_iterator_ &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const binary_heap_point_const_iterator_ &other) const`

### Public Attributes

- `entry_pointer m_p_e`

### Protected Types

- `typedef _Alloc::template rebind< Entry >::other::pointer entry_pointer`

#### 5.223.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

Definition at line 55 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.223.2 Member Typedef Documentation

##### 5.223.2.1 const\_pointer

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef _Alloc::template rebind<value_type>::other::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 77 of file `binary_heap_/point_const_iterator.hpp`.

##### 5.223.2.2 const\_reference

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef _Alloc::template rebind<value_type>::other::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 87 of file `binary_heap_/point_const_iterator.hpp`.

##### 5.223.2.3 difference\_type

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::difference_type
```

Difference type.

Definition at line 65 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.223.2.4 iterator\_category

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```

Category.

Definition at line 62 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 5.223.2.5 pointer

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef _Alloc::template rebind<value_type>::other::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_
Value_Type, Entry, Simple, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 72 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 5.223.2.6 reference

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef _Alloc::template rebind<value_type>::other::reference __gnu_pbds::detail::binary_heap_point_const_iterat
Value_Type, Entry, Simple, _Alloc >::reference
```

Iterator's reference type.

Definition at line 82 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 5.223.2.7 value\_type

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef Value_Type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```

Iterator's value type.

Definition at line 68 of file binary\_heap\_/point\_const\_iterator.hpp.

### 5.223.3 Constructor & Destructor Documentation

#### 5.223.3.1 `binary_heap_point_const_iterator_()` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::binary_heap_point_const_iterator_ ( ) [inline]
```

Default constructor.

Definition at line 95 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.223.3.2 `binary_heap_point_const_iterator_()` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::binary_heap_point_const_iterator_ (
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) [inline]
```

Copy constructor.

Definition at line 99 of file `binary_heap_/point_const_iterator.hpp`.

### 5.223.4 Member Function Documentation

#### 5.223.4.1 `operator!=(())`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator!=(
    const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.223.4.2 `operator*()`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* ( ) const [inline]
```

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.



#### 5.223.4.3 operator->()

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> ( ) const [inline]
```

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

#### 5.223.4.4 operator==()

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >↵
::operator==(
    const binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc > & other
) const [inline]
```

Compares content to a different iterator object.

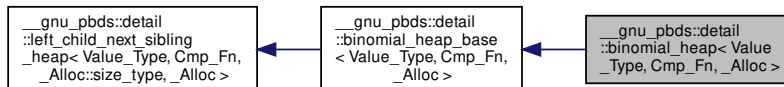
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/point\\_const\\_iterator.hpp](#)

### 5.224 \_\_gnu\_pbds::detail::binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



#### Public Types

- typedef `base_type::allocator_type` **allocator\_type**
- typedef `base_type::cmp_fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binomial\_heap** (const Cmp\_Fn &)
- **binomial\_heap** (const [binomial\\_heap](#) &)
- **iterator begin** ()
- **const\_iterator begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator end** ()
- **const\_iterator end** () const
- void **erase** ([point\\_iterator](#))
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** ([point\\_iterator](#), const\_reference)
- void **pop** ()
- [point\\_iterator](#) **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** ([left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

## Protected Types

- typedef base\_type::node **node**
- typedef \_Alloc::template rebind< [left\\_child\\_next\\_sibling\\_heap\\_node\\_](#)< Value\_Type, \_Alloc::size\_type, \_Alloc >::other **node\_allocator**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It >  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.224.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binomial heap.

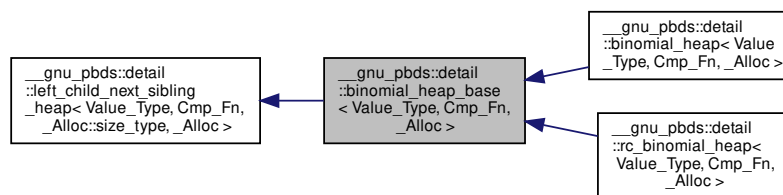
Definition at line 68 of file binomial\_heap\_.hpp.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_.hpp](#)

#### 5.225 \_\_gnu\_pbds::detail::binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &`)
- size\_type **max\_size** () const
- void **modify** (`point_iterator`, const\_reference)
- void **pop** ()
- `point_iterator` **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, `binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &`)
- void **swap** (`left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &`)
- const\_reference **top** () const

## Protected Types

- typedef `base_type::node` **node**
- typedef `_Alloc::template rebind< left_child_next_sibling_heap_node_< Value_Type, _Alloc::size_type, _Alloc >::other` **node\_allocator**
- typedef `base_type::node_const_pointer` **node\_const\_pointer**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef `base_type::node_pointer` **node\_pointer**
- typedef `std::pair< node_pointer, node_pointer >` **node\_pointer\_pair**

### Protected Member Functions

- **binomial\_heap\_base** (const Cmp\_Fn &)
- **binomial\_heap\_base** (const [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It >  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.225.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>  
class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >
```

Base class for binomial heap.

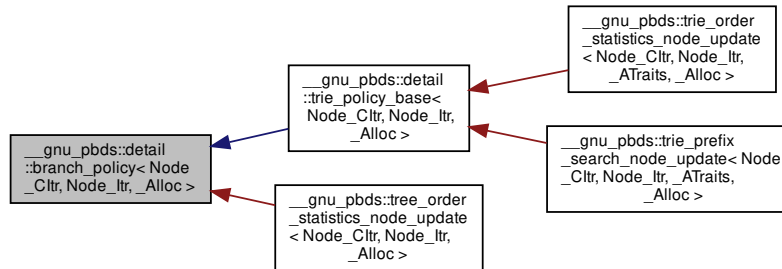
Definition at line 77 of file [binomial\\_heap\\_base\\_.hpp](#).

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_base\\_.hpp](#)

## 5.226 \_\_gnu\_pbds::detail::branch\_policy&lt; Node\_Cltr, Node\_Itr, \_Alloc &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >:



## Protected Types

- typedef rebind\_v::const\_pointer **const\_pointer**
- typedef rebind\_v::const\_reference **const\_reference**
- typedef Node\_Itr::value\_type **it\_type**
- typedef rebind\_k::const\_reference **key\_const\_reference**
- typedef value\_type::first\_type **key\_type**
- typedef remove\_const< key\_type >::type **rkey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef \_Alloc::template rebind< rkey\_type >::other **rebind\_k**
- typedef \_Alloc::template rebind< rcvalue\_type >::other **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef std::iterator\_traits< it\_type >::value\_type **value\_type**

## Protected Member Functions

- virtual it\_type **end** ()=0
- it\_type **end\_iterator** () const

## Static Protected Member Functions

- static key\_const\_reference **extract\_key** (const\_reference r\_val)

## 5.226.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >
```

Primary template, base class for branch structure policies.

Definition at line 52 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 5.227 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

### Protected Types

- `typedef rebind_v::const_pointer` **const\_pointer**
- `typedef rebind_v::const_reference` **const\_reference**
- `typedef Node_Cltr::value_type` **it\_type**
- `typedef rebind_v::const_reference` **key\_const\_reference**
- `typedef value_type` **key\_type**
- `typedef remove_const< value_type >::type` **rcvalue\_type**
- `typedef _Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- `typedef rebind_v::reference` **reference**
- `typedef std::iterator_traits< it_type >::value_type` **value\_type**

### Protected Member Functions

- `virtual it_type` **end** () const =0
- `it_type` **end\_iterator** () const

### Static Protected Member Functions

- `static key_const_reference` **extract\_key** (const\_reference r\_val)

#### 5.227.1 Detailed Description

```
template<typename Node_Cltr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >
```

Specialization for const iterators.

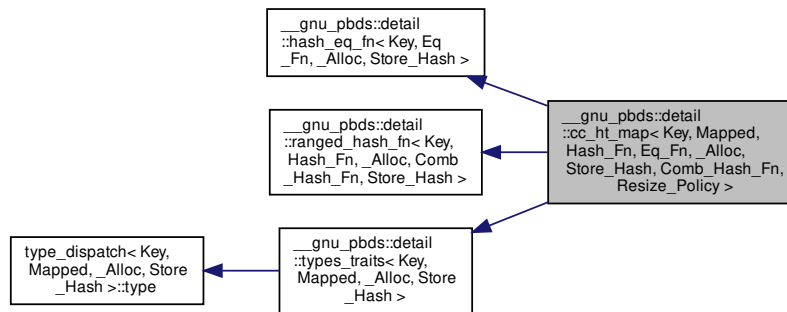
Definition at line 88 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 5.228 `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`:



### Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Hash_Fn` **comb\_hash\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `traits_base::value_type` **value\_type**



## Public Member Functions

- **cc\_ht\_map** (const Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &, const Resize\_Policy &)
- **cc\_ht\_map** (const **cc\_ht\_map**< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It >  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Hash\_Fn & **get\_comb\_hash\_fn** ()
- const Comb\_Hash\_Fn & **get\_comb\_hash\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const
- Hash\_Fn & **get\_hash\_fn** ()
- const Hash\_Fn & **get\_hash\_fn** () const
- Resize\_Policy & **get\_resize\_policy** ()
- const Resize\_Policy & **get\_resize\_policy** () const
- void **initialize** ()
- **std::pair**< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (**cc\_ht\_map**< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- class **const\_iterator\_**
- class **iterator\_**

### 5.228.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >
```

A collision-chaining hash-based container.

#### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Hash_Fn</i>	Combining hash functor. If <code>Hash_Fn</code> is not <code>null_type</code> , then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See <code>Design::Hash-Based Containers::Hash Policies</code> .) Default <code>direct_mask_range_hashing</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_hash_fn`, `detail::types_traits`. (Optional: `detail::debug_↵map_base`.)

Definition at line 139 of file `cc_ht_map.hpp`.

### 5.228.2 Member Enumeration Documentation

#### 5.228.2.1 anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

Definition at line 200 of file `cc_ht_map.hpp`.

### 5.228.3 Member Function Documentation

### 5.228.3.1 empty()

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn,
Resize_Policy >::empty ( ) const [inline]
```

True if size() == 0.

Definition at line 52 of file cc\_ht\_map.hpp.

### 5.228.3.2 get\_comb\_hash\_fn() [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ( )
```

Return current comb\_hash\_fn.

Definition at line 70 of file cc\_ht\_map.hpp.

### 5.228.3.3 get\_comb\_hash\_fn() [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ( ) const
```

Return current const comb\_hash\_fn.

Definition at line 76 of file cc\_ht\_map.hpp.

### 5.228.3.4 get\_eq\_fn() [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn,
Resize_Policy >::get_eq_fn ( )
```

Return current eq\_fn.

Definition at line 58 of file cc\_ht\_map.hpp.

#### 5.228.3.5 `get_eq_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_eq_fn ( ) const
```

Return current const eq\_fn.

Definition at line 64 of file `cc_ht_map.hpp`.

#### 5.228.3.6 `get_hash_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Hash_Fn, Resize_Policy >::get_hash_fn ( )
```

Return current hash\_fn.

Definition at line 46 of file `cc_ht_map.hpp`.

#### 5.228.3.7 `get_hash_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_hash_fn ( ) const
```

Return current const hash\_fn.

Definition at line 52 of file `cc_ht_map.hpp`.

#### 5.228.3.8 `get_resize_policy()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy ( )
```

Return current resize\_policy.

Definition at line 82 of file `cc_ht_map.hpp`.

### 5.228.3.9 `get_resize_policy()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ( ) const
```

Return current const `resize_policy`.

Definition at line 88 of file `cc_ht_map.hpp`.

The documentation for this class was generated from the following file:

- [cc\\_ht\\_map.hpp](#)

## 5.229 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

### Public Types

- typedef `HT_Map::entry` **entry**
- typedef `HT_Map::entry_allocator` **entry\_allocator**
- typedef `__rebind_e::other` **entry\_allocator**
- typedef `entry_allocator::pointer` **entry\_pointer**
- typedef `HT_Map::key_type` **key\_type**

### Public Member Functions

- **cond\_dealtor** (`entry_allocator *p_a`, `entry *p_e`)
- **cond\_dealtor** (`entry_pointer p_e`)
- void **set\_key\_destruct** ()
- void **set\_no\_action** ()
- void **set\_no\_action\_destructor** ()

### Protected Attributes

- bool **m\_key\_destruct**
- `entry_allocator *const` **m\_p\_a**
- `entry *const` **m\_p\_e**

### 5.229.1 Detailed Description

```
template<typename Entry, typename _Alloc>
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
```

Conditional deallocate constructor argument.

Conditional key destructor, `cc_hash`.

Definition at line 50 of file `cond_dealtor.hpp`.

The documentation for this class was generated from the following files:

- [cond\\_dealtor.hpp](#)
- [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#)

## 5.230 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >` Struct Template Reference

### 5.230.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_Tl = null_type>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >
```

Dispatch mechanism, primary template for associative types.

Definition at line 449 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.231 `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >` Struct Template Reference

### Public Types

- typedef `binary_heap< _VTp, Cmp_Fn, _Alloc >` [type](#)

### 5.231.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >
```

Specialization for `binary_heap`.

Definition at line 95 of file `priority_queue_base_dispatch.hpp`.

### 5.231.2 Member Typedef Documentation

#### 5.231.2.1 `type`

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef binary\_heap<\_VTp, Cmp\_Fn, \_Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp,
Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >::type
```

Dispatched type.

Definition at line 99 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

## 5.232 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >` Struct Template Reference

### Public Types

- typedef `binomial_heap<_VTp, Cmp_Fn, _Alloc >` `type`

### 5.232.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >
```

Specialization for `binomial_heap`.

Definition at line 77 of file `priority_queue_base_dispatch.hpp`.

### 5.232.2 Member Typedef Documentation

#### 5.232.2.1 `type`

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp,
Cmp_Fn, _Alloc, binomial_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 81 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `priority_queue_base_dispatch.hpp`

## 5.233 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >` Struct Template Reference

### Public Types

- typedef `pairing_heap<_VTp, Cmp_Fn, _Alloc >` `type`

#### 5.233.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >
```

Specialization for `pairing_heap`.

Definition at line 68 of file `priority_queue_base_dispatch.hpp`.

#### 5.233.2 Member Typedef Documentation

##### 5.233.2.1 `type`

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef pairing\_heap<_VTp, Cmp_Fn, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<_VTp,
Cmp_Fn, _Alloc, pairing\_heap\_tag, null\_type >::type
```

Dispatched type.

Definition at line 72 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 5.234 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type>` Struct Template Reference

##### Public Types

- typedef [rc\\_binomial\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc> [type](#)

#### 5.234.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >
```

Specialization for `rc_binary_heap`.

Definition at line 86 of file `priority_queue_base_dispatch.hpp`.



### 5.234.2 Member Typedef Documentation

#### 5.234.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 90 of file priority\_queue\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 5.235 \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type > Struct Template Reference

#### Public Types

- typedef [thin\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 5.235.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >
```

Specialization for thin\_heap.

Definition at line 104 of file priority\_queue\_base\_dispatch.hpp.

### 5.235.2 Member Typedef Documentation

#### 5.235.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_↵
Fn, _Alloc, thin_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 108 of file priority\_queue\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

## 5.236 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl > Struct` Template Reference

### Public Types

- typedef `cc_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` [type](#)

### 5.236.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash map.

Definition at line 258 of file `container_base_dispatch.hpp`.

### 5.236.2 Member Typedef Documentation

#### 5.236.2.1 `type`

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_d
Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 275 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.237 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl > Struct` Template Reference

### Public Types

- typedef `gp_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` [type](#)

### 5.237.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash map.

Definition at line 303 of file container\_base\_dispatch.hpp.

### 5.237.2 Member Typedef Documentation

#### 5.237.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_
Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 322 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 5.238 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_Tl > Struct Template Reference

#### Public Types

- typedef [lu\\_map](#)< Key, Mapped, at0t, \_Alloc, at1t > [type](#)

#### 5.238.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update map.

Definition at line 107 of file container\_base\_dispatch.hpp.

## 5.238.2 Member Typedef Documentation

### 5.238.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef lu\_map<Key, Mapped, at0t, _Alloc, at1t> \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,
Mapped, _Alloc, list\_update\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 118 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.239 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_Tl > Struct

### Template Reference

#### Public Types

- typedef [ov\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

### 5.239.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree map.

Definition at line 227 of file `container_base_dispatch.hpp`.

## 5.239.2 Member Typedef Documentation

### 5.239.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef ov\_tree\_map<Key, Mapped, at0t, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, Mapped, _Alloc, ov\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 237 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.240 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef [pat\\_trie\\_map](#)< Key, Mapped, at1t, \_Alloc > **type**

#### 5.240.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie map.

Definition at line 139 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.241 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef [rb\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > **type**

#### 5.241.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree map.

Definition at line 165 of file `container_base_dispatch.hpp`.

#### 5.241.2 Member Typedef Documentation

### 5.241.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef rb\_tree\_map<Key, Mapped, at0t, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, Mapped, _Alloc, rb\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 175 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.242 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef [splay\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

### 5.242.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree map.

Definition at line 195 of file `container_base_dispatch.hpp`.

### 5.242.2 Member Typedef Documentation

#### 5.242.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef splay\_tree\_map<Key, Mapped, at0t, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, Mapped, _Alloc, splay\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 206 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.243 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef `cc_ht_set< Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` [type](#)

### 5.243.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash set.

Definition at line 280 of file `container_base_dispatch.hpp`.

### 5.243.2 Member Typedef Documentation

#### 5.243.2.1 `type`

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef cc_ht_set<Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> \_\_gnu\_pbds::detail::container\_base\_dispatch
Key, null\_type, _Alloc, cc\_hash\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 298 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.244 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef `gp_ht_set< Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` [type](#)

## 5.244.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash set.

Definition at line 327 of file `container_base_dispatch.hpp`.

## 5.244.2 Member Typedef Documentation

## 5.244.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 347 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

5.245 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >`

## Struct Template Reference

## Public Types

- typedef `lu_set< Key, null_type, at0t, _Alloc, at1t >` type

## 5.245.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update set.

Definition at line 123 of file `container_base_dispatch.hpp`.



## 5.245.2 Member Typedef Documentation

### 5.245.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 134 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.246 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_Tl > Struct Template Reference

### Public Types

- typedef ov\_tree\_set< Key, null\_type, at0t, at1t, \_Alloc > type

### 5.246.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree set.

Definition at line 242 of file container\_base\_dispatch.hpp.

## 5.246.2 Member Typedef Documentation

### 5.246.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 253 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.247 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl > Struct` Template Reference

### Public Types

- `typedef pat_trie_set< Key, null\_type, at1t, _Alloc > type`

#### 5.247.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie set.

Definition at line 151 of file `container_base_dispatch.hpp`.

#### 5.247.2 Member Typedef Documentation

##### 5.247.2.1 `type`

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef pat_trie_set<Key, null\_type, at1t, _Alloc> \_\_gnu\_pbds::detail::container\_base\_dispatch<
Key, null\_type, _Alloc, pat\_trie\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 160 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.248 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl > Struct` Template Reference

### Public Types

- `typedef rb_tree_set< Key, null\_type, at0t, at1t, _Alloc > type`

#### 5.248.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree set.

Definition at line 180 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 5.249 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_Tl > Struct Template Reference

##### Public Types

- typedef splay\_tree\_set< Key, [null\\_type](#), at0t, at1t, \_Alloc > [type](#)

#### 5.249.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree set.

Definition at line 211 of file container\_base\_dispatch.hpp.

#### 5.249.2 Member Typedef Documentation

##### 5.249.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef splay_tree_set<Key, null\_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null\_type, _Alloc, splay\_tree\_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 222 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 5.250 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

### Public Types

- typedef [direct\\_mask\\_range\\_hashing](#) type

#### 5.250.1 Detailed Description

Primary template, `default_comb_hash_fn`.

Definition at line 80 of file `standard_policies.hpp`.

#### 5.250.2 Member Typedef Documentation

##### 5.250.2.1 type

```
typedef direct\_mask\_range\_hashing \_\_gnu\_pbds::detail::default\_comb\_hash\_fn::type
```

Dispatched type.

Definition at line 83 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.251 `__gnu_pbds::detail::default_eq_fn<Key>` Struct Template Reference

### Public Types

- typedef [std::equal\\_to<Key>](#) type

#### 5.251.1 Detailed Description

```
template<typename Key>  
struct \_\_gnu\_pbds::detail::default\_eq\_fn<Key>
```

Primary template, `default_eq_fn`.

Definition at line 67 of file `standard_policies.hpp`.

### 5.251.2 Member Typedef Documentation

#### 5.251.2.1 type

```
template<typename Key>
typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type
```

Dispatched type.

Definition at line 70 of file standard\_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 5.252 \_\_gnu\_pbds::detail::default\_hash\_fn< Key > Struct Template Reference

#### Public Types

- typedef std::tr1::hash< Key > [type](#)

#### 5.252.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_hash_fn< Key >
```

Primary template, default\_hash\_fn.

Definition at line 59 of file standard\_policies.hpp.

### 5.252.2 Member Typedef Documentation

#### 5.252.2.1 type

```
template<typename Key>
typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type
```

Dispatched type.

Definition at line 62 of file standard\_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.253 `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >` Struct Template Reference

### Public Types

- typedef `cond_type::__type` [type](#)

#### 5.253.1 Detailed Description

```
template<typename Comb_Probe_Fn>
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
```

Primary template, `default_probe_fn`.

Definition at line 117 of file `standard_policies.hpp`.

#### 5.253.2 Member Typedef Documentation

##### 5.253.2.1 `type`

```
template<typename Comb_Probe_Fn >
typedef cond_type::__type \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn >::type
```

Dispatched type.

Definition at line 129 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.254 `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >` Struct Template Reference

### Public Types

- typedef [hash\\_standard\\_resize\\_policy](#)< `size_policy_type`, [trigger](#), `false`, `size_type` > [type](#)

#### 5.254.1 Detailed Description

```
template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
```

Primary template, `default_resize_policy`.

Definition at line 88 of file `standard_policies.hpp`.

## 5.254.2 Member Typedef Documentation

### 5.254.2.1 type

```
template<typename Comb_Hash_Fn>
typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_res
Comb_Hash_Fn >::type
```

Dispatched type.

Definition at line 105 of file standard\_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.255 \_\_gnu\_pbds::detail::default\_trie\_access\_traits< Key > Struct Template Reference

### 5.255.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_trie_access_traits< Key >
```

Primary template, default\_trie\_access\_traits.

Definition at line 135 of file standard\_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.256 \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string< Char, Char\_Traits, std::allocator< char > > > Struct Template Reference

### Public Types

- typedef [trie\\_string\\_access\\_traits< string\\_type >](#) type

### 5.256.1 Detailed Description

```
template<typename Char, typename Char_Traits>
struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >
```

Partial specialization, default\_trie\_access\_traits.

Definition at line 142 of file standard\_policies.hpp.

### 5.256.2 Member Typedef Documentation

#### 5.256.2.1 `type`

```
template<typename Char , typename Char_Traits >
typedef trie\_string\_access\_traits<string\_type> \_\_gnu\_pbds::detail::default\_trie\_access\_traits<
std::basic\_string< Char, Char\_Traits, std::allocator< char > > >::type
```

Dispatched type.

Definition at line 149 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 5.257 `__gnu_pbds::detail::default_update_policy` Struct Reference

### Public Types

- typedef [lu\\_move\\_to\\_front\\_policy](#) `type`

#### 5.257.1 Detailed Description

Default update policy.

Definition at line 109 of file `standard_policies.hpp`.

### 5.257.2 Member Typedef Documentation

#### 5.257.2.1 `type`

```
typedef lu\_move\_to\_front\_policy \_\_gnu\_pbds::detail::default\_update\_policy::type
```

Dispatched type.

Definition at line 112 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)



## 5.258 `__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >` Struct Template Reference

### Public Types

- typedef const\_iterator **const\_reference**
- typedef const\_reference **reference**
- typedef const\_iterator **value\_type**

### 5.258.1 Detailed Description

```
template<typename Key, typename Data, typename _Alloc>
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >
```

Constant node iterator.

Definition at line 52 of file `null_node_metadata.hpp`.

The documentation for this struct was generated from the following file:

- [null\\_node\\_metadata.hpp](#)

## 5.259 `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >` Struct Template Reference

### 5.259.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >
```

Entry compare, primary template.

Definition at line 50 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 5.260 `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >` Struct Template Reference

### Classes

- struct [type](#)

## Public Types

- typedef `__rebind_v::other::const_pointer` **entry**

### 5.260.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>
```

Specialization, false.

Definition at line 62 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 5.261 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type` Struct Reference

Inherits `Cmp_Fn`.

## Public Member Functions

- **type** (const `Cmp_Fn` &other)
- bool **operator()** (entry lhs, entry rhs) const

### 5.261.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type
```

Compare plus entry.

Definition at line 71 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 5.262 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>` Struct Template Reference

## Public Types

- typedef `Cmp_Fn` **type**

### 5.262.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >
```

Specialization, true.

Definition at line 54 of file entry\_cmp.hpp.

### 5.262.2 Member Typedef Documentation

#### 5.262.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type
```

Compare.

Definition at line 57 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 5.263 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw > Struct Template Reference

### 5.263.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >
```

Entry predicate primary class template.

Definition at line 50 of file entry\_pred.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 5.264 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false > Struct Template Reference

### Public Types

- typedef \_\_rebind\_v::other::const\_pointer **entry**

## 5.264.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >
```

Specialization, false.

Definition at line 61 of file entry\_pred.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 5.265 \_\_gnu\_pbds::detail::entry\_pred&lt; \_VTp, Pred, \_Alloc, true &gt; Struct Template Reference

## Public Types

- typedef Pred **type**

## 5.265.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >
```

Specialization, true.

Definition at line 54 of file entry\_pred.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 5.266 \_\_gnu\_pbds::detail::eq\_by\_less&lt; Key, Cmp\_Fn &gt; Struct Template Reference

Inherits Cmp\_Fn.

## Public Member Functions

- bool **operator()** (const Key &r\_lhs, const Key &r\_rhs) const

### 5.266.1 Detailed Description

```
template<typename Key, class Cmp_Fn>
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >
```

Equivalence function.

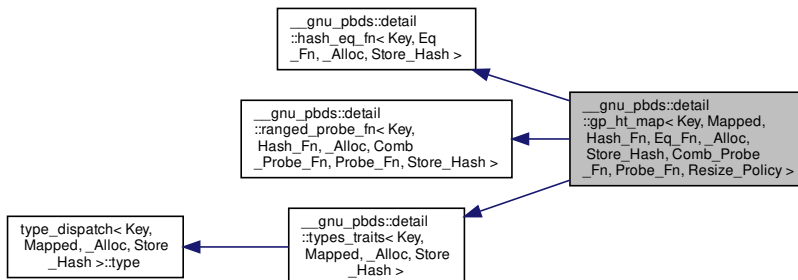
Definition at line 56 of file eq\_by\_less.hpp.

The documentation for this struct was generated from the following file:

- [eq\\_by\\_less.hpp](#)

### 5.267 \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >:



### Public Types

- enum { **store\_hash** }
- typedef \_Alloc **allocator\_type**
- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef [const\\_iterator](#) **const\_iterator**
- typedef traits\_base::const\_pointer **const\_pointer**
- typedef traits\_base::const\_reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef [iterator](#) **iterator**
- typedef traits\_base::key\_const\_pointer **key\_const\_pointer**
- typedef traits\_base::key\_const\_reference **key\_const\_reference**
- typedef traits\_base::key\_pointer **key\_pointer**

- `typedef traits_base::key_reference` **key\_reference**
- `typedef traits_base::key_type` **key\_type**
- `typedef traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- `typedef traits_base::mapped_const_reference` **mapped\_const\_reference**
- `typedef traits_base::mapped_pointer` **mapped\_pointer**
- `typedef traits_base::mapped_reference` **mapped\_reference**
- `typedef traits_base::mapped_type` **mapped\_type**
- `typedef __nothrowcopy::indicator` **no\_throw\_indicator**
- `typedef point_const_iterator` **point\_const\_iterator**
- `typedef point_iterator` **point\_iterator**
- `typedef traits_base::pointer` **pointer**
- `typedef Probe_Fn` **probe\_fn**
- `typedef traits_base::reference` **reference**
- `typedef Resize_Policy` **resize\_policy**
- `typedef _Alloc::size_type` **size\_type**
- `typedef integral_constant< int, Store_Hash >` **store\_extra**
- `typedef traits_base::value_type` **value\_type**

#### Public Member Functions

- **gp\_ht\_map** (const [gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)
- **gp\_ht\_map** (const Hash\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &, const Resize\_Policy &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- `template<typename It >`  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- `template<typename Pred >`  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Probe\_Fn & **get\_comb\_probe\_fn** ()
- const Comb\_Probe\_Fn & **get\_comb\_probe\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const
- Hash\_Fn & **get\_hash\_fn** ()
- const Hash\_Fn & **get\_hash\_fn** () const
- Probe\_Fn & **get\_probe\_fn** ()

- const Probe\_Fn & [get\\_probe\\_fn](#) () const
- Resize\_Policy & [get\\_resize\\_policy](#) ()
- const Resize\_Policy & [get\\_resize\\_policy](#) () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** ([gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)

#### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### Friends

- class
- class **const\_iterator\_**  
**iterator\_**

#### 5.267.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename
Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy
>
```

A general-probing hash-based container.

#### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Probe_Fn</i>	Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> .
<i>Probe_Fn</i>	Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_↵`  
`map_base`.)

Definition at line 142 of file `gp_ht_map.hpp`.

## 5.267.2 Member Enumeration Documentation

### 5.267.2.1 anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

Definition at line 208 of file `gp_ht_map.hpp`.

## 5.267.3 Member Function Documentation

### 5.267.3.1 `empty()`

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
bool \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_↵
Fn, Probe_Fn, Resize_Policy >::empty ( ) const [inline]
```

True if `size() == 0`.

Definition at line 58 of file `gp_ht_map.hpp`.

### 5.267.3.2 `get_comb_probe_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Comb_Probe_Fn & \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ( )
```

Return current `comb_probe_fn`.

Definition at line 82 of file `gp_ht_map.hpp`.



**5.267.3.3** `get_comb_probe_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ( ) const
```

Return current const comb\_probe\_fn.

Definition at line 88 of file gp\_ht\_map.hpp.

**5.267.3.4** `get_eq_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ( )
```

Return current eq\_fn.

Definition at line 58 of file gp\_ht\_map.hpp.

**5.267.3.5** `get_eq_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ( ) const
```

Return current const eq\_fn.

Definition at line 64 of file gp\_ht\_map.hpp.

**5.267.3.6** `get_hash_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ( )
```

Return current hash\_fn.

Definition at line 46 of file gp\_ht\_map.hpp.

### 5.267.3.7 `get_hash_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ( ) const
```

Return current const hash\_fn.

Definition at line 52 of file gp\_ht\_map.hpp.

### 5.267.3.8 `get_probe_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ( )
```

Return current probe\_fn.

Definition at line 70 of file gp\_ht\_map.hpp.

### 5.267.3.9 `get_probe_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ( ) const
```

Return current const probe\_fn.

Definition at line 76 of file gp\_ht\_map.hpp.

### 5.267.3.10 `get_resize_policy()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ( )
```

Return current resize\_policy.

Definition at line 94 of file gp\_ht\_map.hpp.

### 5.267.3.11 `get_resize_policy()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ( ) const
```

Return current const `resize_policy`.

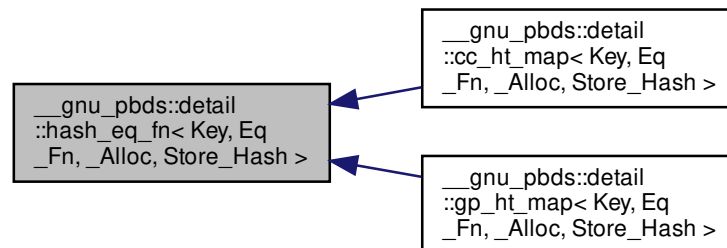
Definition at line 100 of file `gp_ht_map.hpp`.

The documentation for this class was generated from the following file:

- [gp\\_ht\\_map.hpp](#)

## 5.268 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`:



### 5.268.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >
```

Primary template.

Definition at line 54 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

## 5.269 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >` Struct Template Reference

Inherits `Eq_Fn`.

### Public Types

- typedef `Eq_Fn` **eq\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**

### Public Member Functions

- **hash\_eq\_fn** (const `Eq_Fn` &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, key\_const\_reference r\_rhs\_key) const
- void **swap** (const [hash\\_eq\\_fn](#) &other)

#### 5.269.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
```

Specialization 1 - The client requests that hash values not be stored.

Definition at line 58 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

## 5.270 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >` Struct Template Reference

Inherits `Eq_Fn`.

### Public Types

- typedef `Eq_Fn` **eq\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

### Public Member Functions

- **hash\_eq\_fn** (const `Eq_Fn` &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, size\_type lhs\_hash, key\_const\_reference r\_rhs\_key, size\_type rhs\_hash) const
- void **swap** (const [hash\\_eq\\_fn](#) &other)

#### 5.270.1 Detailed Description

```
template<typename Key, class Eq_Fn, class _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >
```

Specialization 2 - The client requests that hash values be stored.

Definition at line 81 of file hash\_eq\_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

### 5.271 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size > Class Template Reference

#### 5.271.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

Definition at line 50 of file hash\_load\_check\_resize\_trigger\_size\_base.hpp.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

### 5.272 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, true > Class Template Reference

#### Protected Types

- typedef Size\_Type **size\_type**

#### Protected Member Functions

- size\_type **get\_size** () const
- void **set\_size** (size\_type size)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger\\_size\\_base](#) &other)

## 5.272.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >
```

Specializations.

Definition at line 54 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

5.273 `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >`  
Class Template Reference

Inherits `Cmp_Fn`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `left_child_next_sibling_heap_const_iterator< node, _Alloc >` **const\_iterator**
- typedef `__rebind_v::other::const_pointer` **const\_pointer**
- typedef `__rebind_v::other::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `const_iterator` **iterator**
- typedef `left_child_next_sibling_heap_node_point_const_iterator< node, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `__rebind_v::other::pointer` **pointer**
- typedef `__rebind_v::other::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `left_child_next_sibling_heap` (const `Cmp_Fn` &)
- `left_child_next_sibling_heap` (const `left_child_next_sibling_heap` &)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- `size_type` **max\_size** () const
- `size_type` **size** () const
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `Node_Metadata`, `_Alloc` > &)

### Protected Types

- typedef node\_allocator::value\_type **node**
- typedef \_Alloc::template rebind< [left\\_child\\_next\\_sibling\\_heap\\_node](#)< Value\_Type, Node\_Metadata, \_Alloc >>::other **node\_allocator**
- typedef node\_allocator::const\_pointer **node\_const\_pointer**
- typedef Node\_Metadata **node\_metadata**
- typedef node\_allocator::pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- template<typename Pred >  
node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.273.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>  
class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >
```

Base class for a basic heap.

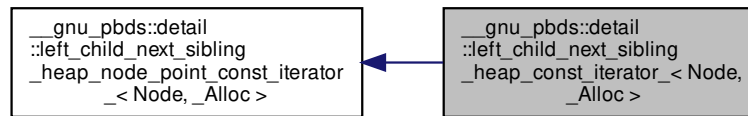
Definition at line 90 of file [left\\_child\\_next\\_sibling\\_heap.hpp](#).

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap.hpp](#)

## 5.274 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



### Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

### Public Member Functions

- `left_child_next_sibling_heap_const_iterator_` (`node_pointer p_nd`)
- `left_child_next_sibling_heap_const_iterator_` ()
- `left_child_next_sibling_heap_const_iterator_` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`)
- `bool operator!=` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`) `const`
- `bool operator!=` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`
- `const_reference operator*` () `const`
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & operator++` ()
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > operator++` (`int`)
- `const_pointer operator->` () `const`
- `bool operator==` (`const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other`) `const`
- `bool operator==` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`

### Public Attributes

- `node_pointer m_p_nd`



### 5.274.1 Detailed Description

```
template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

Definition at line 60 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

### 5.274.2 Member Typedef Documentation

#### 5.274.2.1 const\_pointer

```
template<typename Node , typename _Alloc >
typedef base_type::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 81 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 5.274.2.2 const\_reference

```
template<typename Node , typename _Alloc >
typedef base_type::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 87 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 5.274.2.3 difference\_type

```
template<typename Node , typename _Alloc >
typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::difference_type
```

Difference type.

Definition at line 72 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 5.274.2.4 iterator\_category

```
template<typename Node , typename _Alloc >
typedef std::forward\_iterator\_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::iterator_category
```

Category.

Definition at line 69 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 5.274.2.5 pointer

```
template<typename Node , typename _Alloc >
typedef base\_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 78 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 5.274.2.6 reference

```
template<typename Node , typename _Alloc >
typedef base\_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::reference
```

Iterator's reference type.

Definition at line 84 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 5.274.2.7 value\_type

```
template<typename Node , typename _Alloc >
typedef base\_type::value\_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::value_type
```

Iterator's value type.

Definition at line 75 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

### 5.274.3 Constructor & Destructor Documentation

#### 5.274.3.1 left\_child\_next\_sibling\_heap\_const\_iterator\_() [1/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_co
( ) [inline]
```

Default constructor.

Definition at line 96 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

#### 5.274.3.2 left\_child\_next\_sibling\_heap\_const\_iterator\_() [2/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_co
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other ) [inline]
```

Copy constructor.

Definition at line 101 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

### 5.274.4 Member Function Documentation

#### 5.274.4.1 operator!=(()) [1/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!=
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other ) const
[inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 111 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

#### 5.274.4.2 operator!=(()) [2/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other ) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 137 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.274.4.3 operator\*()

```
template<typename Node , typename _Alloc >
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* ( ) const [inline], [inherited]
```

Access.

Definition at line 124 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.274.4.4 operator->()

```
template<typename Node , typename _Alloc >
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> ( ) const [inline], [inherited]
```

Access.

Definition at line 116 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.274.4.5 operator==( ) [1/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other ) const
[inline]
```

Compares content to a different iterator object.

Definition at line 106 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

#### 5.274.4.6 operator==( ) [2/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other ) const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 132 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/const\\_iterator.hpp](#)

## 5.275 `__gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value, _Metadata, _Alloc >` Struct Template Reference

### Public Types

- typedef `_Metadata` **metadata\_type**
- typedef `_Alloc::template rebind< this\_type >::other::pointer` **node\_pointer**
- typedef `_Alloc::size_type` **size\_type**
- typedef `_Value` **value\_type**

### Public Attributes

- `metadata_type` **m\_metadata**
- `node_pointer` **m\_p\_l\_child**
- `node_pointer` **m\_p\_next\_sibling**
- `node_pointer` **m\_p\_prev\_or\_parent**
- `value_type` **m\_value**

#### 5.275.1 Detailed Description

```
template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value, _Metadata, _Alloc >
```

Node.

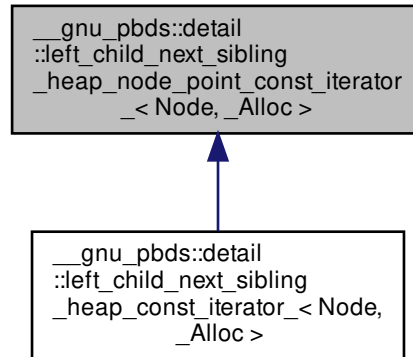
Definition at line 50 of file `left_child_next_sibling_heap_/node.hpp`.

The documentation for this struct was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/node.hpp](#)

## 5.276 `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`:



### Public Types

- `typedef _Alloc::template rebind< value_type >::other::const_pointer const_pointer`
- `typedef _Alloc::template rebind< value_type >::other::const_reference const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef _Alloc::template rebind< value_type >::other::pointer pointer`
- `typedef _Alloc::template rebind< value_type >::other::reference reference`
- `typedef Node::value_type value_type`

### Public Member Functions

- `left_child_next_sibling_heap_node_point_const_iterator_ (node_pointer p_nd)`
- `left_child_next_sibling_heap_node_point_const_iterator_ ()`
- `left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other)`
- `bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`

### Public Attributes

- `node_pointer m_p_nd`

## Protected Types

- typedef `_Alloc::template rebind< Node >::other::pointer` **node\_pointer**

### 5.276.1 Detailed Description

```
template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >
```

Const point-type iterator.

Definition at line 61 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

### 5.276.2 Member Typedef Documentation

#### 5.276.2.1 const\_pointer

```
template<typename Node , typename _Alloc >
typedef _Alloc::template rebind< value_type>::other::const_pointer __gnu_pbds::detail::left_child_next_sibling_h
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 86 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 5.276.2.2 const\_reference

```
template<typename Node , typename _Alloc >
typedef _Alloc::template rebind< value_type>::other::const_reference __gnu_pbds::detail::left_child_next_sibling
Node, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 98 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 5.276.2.3 difference\_type

```
template<typename Node , typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterat
Node, _Alloc >::difference_type
```

Difference type.

Definition at line 71 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 5.276.2.4 iterator\_category

```
template<typename Node , typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::iterator_category
```

Category.

Definition at line 68 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.276.2.5 pointer

```
template<typename Node , typename _Alloc >
typedef _Alloc::template rebind< value_type >::other::pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 80 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.276.2.6 reference

```
template<typename Node , typename _Alloc >
typedef _Alloc::template rebind< value_type >::other::reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::reference
```

Iterator's reference type.

Definition at line 92 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.276.2.7 value\_type

```
template<typename Node , typename _Alloc >
typedef Node::value_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::value_type
```

Iterator's value type.

Definition at line 74 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

### 5.276.3 Constructor & Destructor Documentation



#### 5.276.3.1 left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_() [1/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >↔
::left_child_next_sibling_heap_node_point_const_iterator_ ( ) [inline]
```

Default constructor.

Definition at line 106 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.276.3.2 left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_() [2/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >↔
::left_child_next_sibling_heap_node_point_const_iterator_ (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other ) [inline]
```

Copy constructor.

Definition at line 111 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

### 5.276.4 Member Function Documentation

#### 5.276.4.1 operator!=(())

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!=(
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other ) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 137 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 5.276.4.2 operator\*()

```
template<typename Node , typename _Alloc >
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* ( ) const [inline]
```

Access.

Definition at line 124 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

5.276.4.3 `operator->()`

```
template<typename Node , typename _Alloc >
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> ( ) const [inline]
```

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

5.276.4.4 `operator==()`

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/point\\_const\\_iterator.hpp](#)

5.277 `__gnu_pbds::detail::lu_counter_metadata< Size_Type >` Class Template Reference

## Public Types

- typedef `Size_Type` **size\_type**

## Friends

- class `lu_counter_policy_base< size_type >`

## 5.277.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

Definition at line 51 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

## 5.278 `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >` Class Template Reference

### Protected Types

- typedef `Size_Type` **`size_type`**

### Protected Member Functions

- [lu\\_counter\\_metadata](#)< `size_type` > **`operator()`** (`size_type` `max_size`) `const`
- `template<typename Metadata_Reference >`  
`bool` **`operator()`** (`Metadata_Reference` `r_data`, `size_type` `m_max_count`) `const`

### 5.278.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >
```

Base class for list-update counter policy.

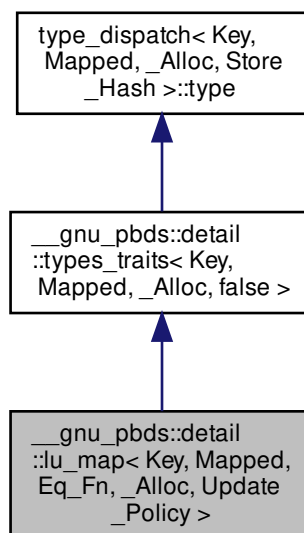
Definition at line 46 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

## 5.279 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `Update_Policy::metadata_type` **update\_metadata**
- typedef `Update_Policy` **update\_policy**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **lu\_map** (const `lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` &)
- `template<typename It >`  
**lu\_map** (It, It)
- iterator **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- `const_iterator` **end** () const
- bool **erase** (key\_const\_reference)
- `template<typename Pred >`  
`size_type` **erase\_if** (Pred)
- `point_iterator` **find** (key\_const\_reference r\_key)
- `point_const_iterator` **find** (key\_const\_reference r\_key) const
- `std::pair< point_iterator, bool >` **insert** (const\_reference)
- `size_type` **max\_size** () const
- `mapped_reference` **operator[]** (key\_const\_reference r\_key)
- `size_type` **size** () const
- void **swap** (`lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Protected Member Functions

- template<typename It >  
void **copy\_from\_range** (It, It)

## Friends

- class **const\_iterator\_**
- class **iterator\_**

## 5.279.1 Detailed Description

```
template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >
```

list-based (with updates) associative container. Skip to the lu, my darling.

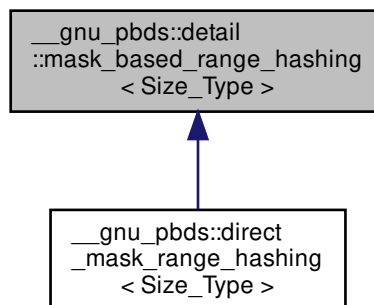
Definition at line 91 of file lu\_map\_.hpp.

The documentation for this class was generated from the following file:

- [lu\\_map\\_.hpp](#)

## 5.280 \_\_gnu\_pbds::detail::mask\_based\_range\_hashing&lt; Size\_Type &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::mask\_based\_range\_hashing< Size\_Type >:



## Protected Types

- typedef `Size_Type` **size\_type**

## Protected Member Functions

- void **notify\_resized** (`size_type` size)
- `size_type` **range\_hash** (`size_type` hash) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

## 5.280.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >
```

Range hashing policy.

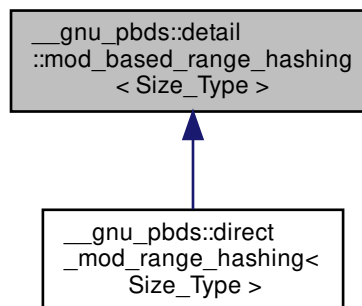
Definition at line 50 of file `mask_based_range_hashing.hpp`.

The documentation for this class was generated from the following file:

- [mask\\_based\\_range\\_hashing.hpp](#)

5.281 `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >`:



## Protected Types

- typedef `Size_Type` **size\_type**

### Protected Member Functions

- void **notify\_resized** (size\_type s)
- size\_type **range\_hash** (size\_type s) const
- void **swap** ([mod\\_based\\_range\\_hashing](#) &other)

#### 5.281.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >
```

Mod based range hashing.

Definition at line 50 of file [mod\\_based\\_range\\_hashing.hpp](#).

The documentation for this class was generated from the following file:

- [mod\\_based\\_range\\_hashing.hpp](#)

### 5.282 \_\_gnu\_pbds::detail::no\_throw\_copies< Key, Mapped > Struct Template Reference

#### Public Types

- typedef integral\_constant< int, \_\_simple > **indicator**

#### Static Public Attributes

- static const bool **\_\_simple**

#### 5.282.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
```

Primary template.

Definition at line 61 of file [types\\_traits.hpp](#).

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

5.283 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference

## Public Types

- `typedef integral_constant< int, is_simple< Key >::value > indicator`

## 5.283.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
```

Specialization.

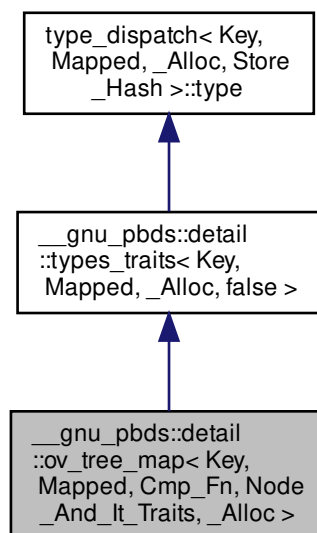
Definition at line 70 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

5.284 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`:





## Classes

- class [cond\\_dtor](#)

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef [ov\\_tree\\_tag](#) **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `traits_type::node_update` **node\_update**
- typedef `const_pointer` **point\_const\_iterator**
- typedef `pointer` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **ov\_tree\_map** (const `Cmp_Fn` &)
- **ov\_tree\_map** (const `Cmp_Fn` &, const `node_update` &)
- **ov\_tree\_map** (const [ov\\_tree\\_map](#)< `Key`, `Mapped`, `Cmp_Fn`, `Node_And_It_Traits`, `_Alloc` > &)
- iterator **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- `template<typename It >`  
void **copy\_from\_range** (`It`, `It`)
- bool **empty** () const
- iterator **end** ()

- `const_iterator end` () const
- `bool erase` (key\_const\_reference)
- `iterator erase` (iterator it)
- `template<typename Pred > size_type erase_if` (Pred)
- `point_iterator find` (key\_const\_reference r\_key)
- `point_const_iterator find` (key\_const\_reference r\_key) const
- `Cmp_Fn & get_cmp_fn` ()
- `const Cmp_Fn & get_cmp_fn` () const
- `std::pair< point_iterator, bool > insert` (const\_reference r\_value)
- `void join` (`ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- `point_iterator lower_bound` (key\_const\_reference r\_key)
- `point_const_iterator lower_bound` (key\_const\_reference r\_key) const
- `size_type max_size` () const
- `node_const_iterator node_begin` () const
- `node_iterator node_begin` ()
- `node_const_iterator node_end` () const
- `node_iterator node_end` ()
- `mapped_reference operator[]` (key\_const\_reference r\_key)
- `size_type size` () const
- `void split` (key\_const\_reference, `ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- `void swap` (`ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &`)
- `point_iterator upper_bound` (key\_const\_reference r\_key)
- `point_const_iterator upper_bound` (key\_const\_reference r\_key) const

#### Public Attributes

- `no_throw_indicator m_no_throw_copies_indicator`
- `store_extra m_store_extra_indicator`

#### 5.284.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Ordered-vector tree associative-container.

Definition at line 106 of file `ov_tree_map.hpp`.

#### 5.284.2 Member Function Documentation

**5.284.2.1 node\_begin()** [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_r
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const [inline]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

Definition at line 45 of file ov\_tree\_map.hpp.

**5.284.2.2 node\_begin()** [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) [inline]
```

Returns a node\_iterator corresponding to the node at the root of the tree.

Definition at line 57 of file ov\_tree\_map.hpp.

**5.284.2.3 node\_end()** [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_r
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 51 of file ov\_tree\_map.hpp.

**5.284.2.4 node\_end()** [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) [inline]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 63 of file ov\_tree\_map.hpp.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map.hpp](#)

5.285 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

Public Member Functions

- **cond\_dtor** (value\_vector a\_vec, iterator &r\_last\_it, Size\_Type total\_size)
- void **set\_no\_action** ()

Protected Attributes

- value\_vector **m\_a\_vec**
- const Size\_Type **m\_max\_size**
- bool **m\_no\_action**
- iterator & **m\_r\_last\_it**

5.285.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >
```

Conditional destructor.

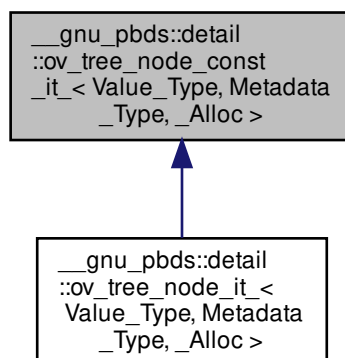
Definition at line 182 of file `ov_tree_map.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map.hpp](#)

5.286 `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >`:



## Public Types

- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **const\_reference**
- typedef `trivial_iterator_difference_type` **difference\_type**
- typedef `trivial_iterator_tag` **iterator\_category**
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` **metadata\_const\_reference**
- typedef `Metadata_Type` **metadata\_type**
- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **reference**
- typedef `_Alloc::template rebind< Value_Type >::other::const_pointer` **value\_type**

## Public Member Functions

- **ov\_tree\_node\_const\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_metadata\_pointer p\_metadata=0)
- `this_type get_l_child` () const
- metadata\_const\_reference **get\_metadata** () const
- `this_type get_r_child` () const
- bool **operator!=** (const `this_type` &other) const
- const\_reference **operator\*** () const
- bool **operator==** (const `this_type` &other) const

## Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

## Protected Types

- typedef `_Alloc::template rebind< Metadata_Type >::other::const_pointer` **const\_metadata\_pointer**
- typedef `_Alloc::template rebind< Value_Type >::other::const_pointer` **const\_pointer**
- typedef `_Alloc::template rebind< Value_Type >::other::pointer` **pointer**
- typedef `ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` **this\_type**

## Static Protected Member Functions

- `template<typename Ptr >`  
static Ptr **mid\_pointer** (Ptr p\_begin, Ptr p\_end)

## 5.286.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >
```

Const node reference.

Definition at line 57 of file `ov_tree_map_/node_iterators.hpp`.

## 5.286.2 Member Function Documentation

5.286.2.1 `get_l_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_l_↵
l_child ( ) const [inline]
```

Returns the node iterator associated with the left node.

Definition at line 142 of file `ov_tree_map_/node_iterators.hpp`.

5.286.2.2 `get_r_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↵
r_child ( ) const [inline]
```

Returns the node iterator associated with the right node.

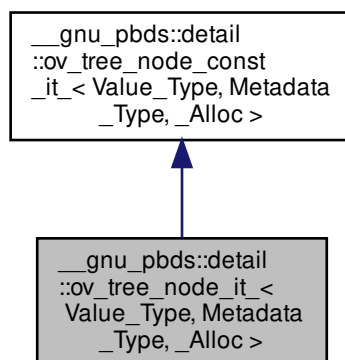
Definition at line 158 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

5.287 `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`:



## Public Types

- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer` **const\_reference**
- typedef `trivial_iterator_difference_type` **difference\_type**
- typedef `trivial_iterator_tag` **iterator\_category**
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` **metadata\_const\_reference**
- typedef `Metadata_Type` **metadata\_type**
- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer` **reference**
- typedef `_Alloc::template rebind< Value_Type >::other::pointer` **value\_type**

## Public Member Functions

- **ov\_tree\_node\_it** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_metadata\_pointer p\_metadata=0)
- **ov\_tree\_node\_it\_get\_l\_child** () const
- metadata\_const\_reference **get\_metadata** () const
- **ov\_tree\_node\_it\_get\_r\_child** () const
- bool **operator!=** (const **this\_type** &other) const
- reference **operator\*** () const
- bool **operator==** (const **this\_type** &other) const

## Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

## Static Protected Member Functions

- template<typename Ptr >  
static Ptr **mid\_pointer** (Ptr p\_begin, Ptr p\_end)

### 5.287.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >
```

Node reference.

Definition at line 204 of file `ov_tree_map_/node_iterators.hpp`.

### 5.287.2 Member Function Documentation

5.287.2.1 `get_l_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↔
_l_child ( ) const [inline]
```

Returns the node reference associated with the left node.

Definition at line 252 of file `ov_tree_map_/node_iterators.hpp`.

5.287.2.2 `get_r_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↔
_r_child ( ) const [inline]
```

Returns the node reference associated with the right node.

Definition at line 268 of file `ov_tree_map_/node_iterators.hpp`.

5.287.2.3 `operator*()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* (
) const [inline]
```

Access.

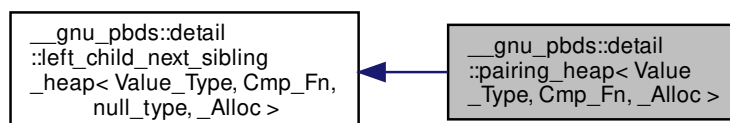
Definition at line 247 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

5.288 `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`:





### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- **pairing\_heap** (const `Cmp_Fn` &)
- **pairing\_heap** (const `pairing_heap` &)
- **iterator** **begin** ()
- **const\_iterator** **begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator** **end** ()
- **const\_iterator** **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`pairing_heap` &)
- size\_type **max\_size** () const
- void **modify** (`point_iterator`, const\_reference)
- void **pop** ()
- `point_iterator` **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, `pairing_heap` &)
- void **swap** (`pairing_heap` &)
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `null_type`, `_Alloc` > &)
- const\_reference **top** () const

### Protected Types

- typedef `node_allocator::value_type` **node**
- typedef `_Alloc::template rebind< left_child_next_sibling_heap_node_< Value_Type, null_type, _Alloc > >::other` **node\_allocator**
- typedef `node_allocator::const_pointer` **node\_const\_pointer**
- typedef `null_type` **node\_metadata**
- typedef `std::pair< node_pointer, node_pointer >` **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It >  
void **copy\_from\_range** (It, It)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.288.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>  
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
```

Pairing heap.

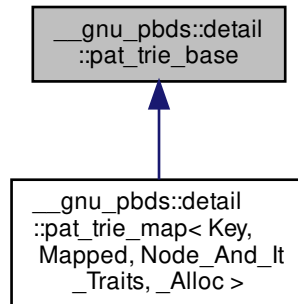
Definition at line 77 of file `pairing_heap_.hpp`.

The documentation for this class was generated from the following file:

- [pairing\\_heap\\_.hpp](#)

## 5.289 `__gnu_pbds::detail::pat_trie_base` Struct Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base`:



### Classes

- class [\\_CIter](#)
- struct [\\_Head](#)
- struct [\\_Inode](#)
- class [\\_Iter](#)
- struct [\\_Leaf](#)
- struct [\\_Metadata](#)
- struct [\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_Node\\_base](#)
- class [\\_Node\\_citer](#)
- class [\\_Node\\_iter](#)

### Public Types

- enum [node\\_type](#) { [i\\_node](#), [leaf\\_node](#), [head\\_node](#) }

#### 5.289.1 Detailed Description

Base type for PATRICIA trees.

Definition at line 51 of file `pat_trie_base.hpp`.

#### 5.289.2 Member Enumeration Documentation

5.289.2.1 `node_type`

```
enum __gnu_pbds::detail::pat_trie_base::node_type
```

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

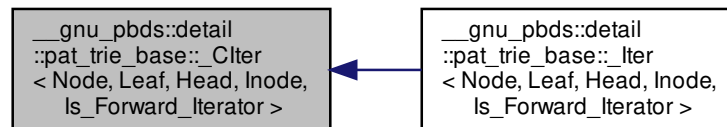
Definition at line 58 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

5.290 `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



## Public Types

- `typedef _Alloc::template rebind< Head > __rebind_h`
- `typedef _Alloc::template rebind< Inode > __rebind_in`
- `typedef _Alloc::template rebind< Leaf > __rebind_l`
- `typedef _Alloc::template rebind< Node > __rebind_n`
- `typedef allocator_type _Alloc`
- `typedef Node::allocator_type allocator_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef __rebind_h::other::pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef __rebind_in::other::pointer inode_pointer`
- `typedef std::bidirectional\_iterator\_tag iterator_category`
- `typedef __rebind_l::other::const_pointer leaf_const_pointer`
- `typedef __rebind_l::other::pointer leaf_pointer`
- `typedef __rebind_n::other::pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef Node::type_traits type_traits`
- `typedef type_traits::value_type value_type`

## Public Member Functions

- **\_Clter** (node\_pointer p\_nd=0)
- **\_Clter** (const **\_Clter**< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- bool **operator!=** (const **\_Clter** &other) const
- bool **operator!=** (const **\_Clter**< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const
- const\_reference **operator\*** () const
- **\_Clter** & **operator++** ()
- **\_Clter** **operator++** (int)
- **\_Clter** & **operator--** ()
- **\_Clter** **operator--** (int)
- const\_pointer **operator->** () const
- **\_Clter** & **operator=** (const **\_Clter** &other)
- **\_Clter** & **operator=** (const **\_Clter**< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- bool **operator==** (const **\_Clter** &other) const
- bool **operator==** (const **\_Clter**< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const

## Public Attributes

- node\_pointer **m\_p\_nd**

## Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

## Static Protected Member Functions

- static node\_pointer **get\_larger\_sibling** (node\_pointer p\_nd)
- static node\_pointer **get\_smaller\_sibling** (node\_pointer p\_nd)
- static leaf\_pointer **leftmost\_descendant** (node\_pointer p\_nd)
- static leaf\_pointer **rightmost\_descendant** (node\_pointer p\_nd)

### 5.290.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

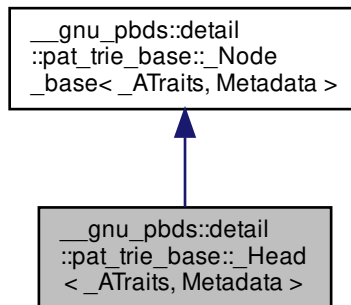
Definition at line 487 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

5.291 `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>`:



## Public Types

- `typedef _Alloc::template rebind<_ATraits> __rebind_at`
- `typedef _Alloc::template rebind<_Node_base> __rebind_n`
- `typedef _ATraits::const_iterator a_const_iterator`
- `typedef __rebind_at::other::const_pointer a_const_pointer`
- `typedef _ATraits access_traits`
- `typedef _Alloc allocator_type`
- `typedef _Node_base<_ATraits, Metadata> base_type`
- `typedef base_type::node_pointer node_pointer`
- `typedef base_type::type_traits type_traits`

## Public Attributes

- `node_pointer m_p_max`
- `node_pointer m_p_min`
- `node_pointer m_p_parent`
- `const node_type m_type`

## 5.291.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>
```

Head node for PATRICIA tree.

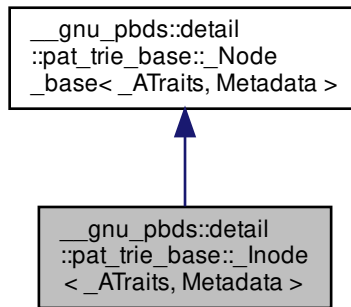
Definition at line 131 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.292 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>`:



### Classes

- struct [const\\_iterator](#)
- struct [iterator](#)

### Public Types

- enum { **arr\_size** }
- typedef `_Alloc::template rebind<_ATraits>` **\_\_rebind\_at**
- typedef `_Alloc::template rebind<node_pointer>::other` **\_\_rebind\_np**
- typedef `base_type::allocator_type` **\_Alloc**
- typedef `base_type::access_traits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `_Node_base<_ATraits, Metadata>` **base\_type**
- typedef `__rebind_np::pointer` **node\_pointer\_pointer**
- typedef `__rebind_np::reference` **node\_pointer\_reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `base_type::type_traits` **type\_traits**
- typedef `type_traits::value_type` **value\_type**

## Public Member Functions

- `_Inode` (size\_type, const a\_const\_iterator)
- node\_pointer **add\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- [const\\_iterator](#) **begin** () const
- [iterator](#) **begin** ()
- [const\\_iterator](#) **end** () const
- [iterator](#) **end** ()
- [iterator](#) **get\_child\_it** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_const\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer) const
- size\_type **get\_e\_ind** () const
- node\_const\_pointer **get\_join\_child** (node\_const\_pointer, a\_const\_pointer) const
- node\_pointer **get\_join\_child** (node\_pointer, a\_const\_pointer)
- node\_pointer **get\_lower\_bound\_child\_node** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer)
- leaf\_pointer **leftmost\_descendant** ()
- leaf\_const\_pointer **leftmost\_descendant** () const
- a\_const\_iterator **pref\_b\_it** () const
- a\_const\_iterator **pref\_e\_it** () const
- void **remove\_child** (node\_pointer)
- void **remove\_child** ([iterator](#))
- void **replace\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- leaf\_pointer **rightmost\_descendant** ()
- leaf\_const\_pointer **rightmost\_descendant** () const
- bool **should\_be\_mine** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer) const
- void **update\_prefixes** (a\_const\_pointer)

## Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

## 5.292.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>
```

Internal node type, PATRICIA tree.

Definition at line 211 of file `pat_trie_base.hpp`.

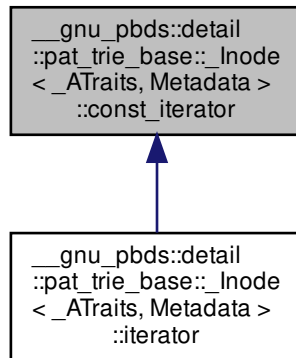
The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)



## 5.293 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator` Struct Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator`:



### Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

### Public Member Functions

- **const\_iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- bool **operator!=** (`const const_iterator &other`) const
- `node_const_pointer` **operator\*** () const
- `const_iterator &` **operator++** ()
- `const_iterator` **operator++** (int)
- `const node_pointer_pointer` **operator->** () const
- bool **operator==** (`const const_iterator &other`) const

### Public Attributes

- `node_pointer_pointer` **m\_p\_p\_cur**
- `node_pointer_pointer` **m\_p\_p\_end**

## 5.293.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::const_iterator
```

Constant child iterator.

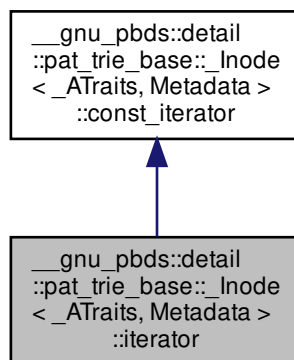
Definition at line 255 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.294 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode&lt;\_ATraits, Metadata &gt;::iterator Struct Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata >::iterator:



## Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

## Public Member Functions

- **iterator** (node\_pointer\_pointer p\_p\_cur=0, node\_pointer\_pointer p\_p\_end=0)
- bool **operator!=** (const [const\\_iterator](#) &other) const
- bool **operator!=** (const [iterator](#) &other) const
- node\_const\_pointer **operator\*** () const
- node\_pointer **operator\*** ()
- [iterator](#) & **operator++** ()
- [iterator](#) **operator++** (int)
- const node\_pointer\_pointer **operator->** () const
- node\_pointer\_pointer **operator->** ()
- bool **operator==** (const [const\\_iterator](#) &other) const
- bool **operator==** (const [iterator](#) &other) const

## Public Attributes

- node\_pointer\_pointer **m\_p\_p\_cur**
- node\_pointer\_pointer **m\_p\_p\_end**

## 5.294.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator
```

Child iterator.

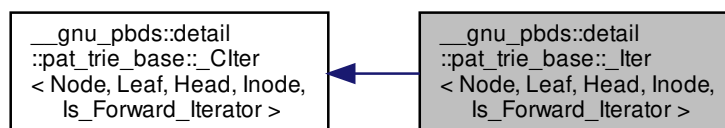
Definition at line 320 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.295 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Iter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Iter< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >:



## Public Types

- `typedef _Alloc::template rebind< Head > __rebind_h`
- `typedef _Alloc::template rebind< Inode > __rebind_in`
- `typedef _Alloc::template rebind< Leaf > __rebind_l`
- `typedef _Alloc::template rebind< Node > __rebind_n`
- `typedef allocator_type _Alloc`
- `typedef base_type::allocator_type allocator_type`
- `typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef base_type::head_pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef base_type::inode_pointer inode_pointer`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef base_type::leaf_const_pointer leaf_const_pointer`
- `typedef base_type::leaf_pointer leaf_pointer`
- `typedef base_type::node_pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

## Public Member Functions

- `_Iter (node_pointer p_nd=0)`
- `_Iter (const _Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other)`
- `bool operator!= (const _CIter &other) const`
- `bool operator!= (const _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other) const`
- `reference operator* () const`
- `_Iter & operator++ ()`
- `_Iter operator++ (int)`
- `_Iter & operator-- ()`
- `_Iter operator-- (int)`
- `pointer operator-> () const`
- `_Iter & operator= (const _Iter &other)`
- `_Iter & operator= (const _Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other)`
- `bool operator== (const _CIter &other) const`
- `bool operator== (const _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > &other) const`

## Public Attributes

- `node_pointer m_p_nd`

## Protected Member Functions

- `void dec (false_type)`
- `void dec (true_type)`
- `void inc (false_type)`
- `void inc (true_type)`

## Static Protected Member Functions

- static node\_pointer **get\_larger\_sibling** (node\_pointer p\_nd)
- static node\_pointer **get\_smaller\_sibling** (node\_pointer p\_nd)
- static leaf\_pointer **leftmost\_descendant** (node\_pointer p\_nd)
- static leaf\_pointer **rightmost\_descendant** (node\_pointer p\_nd)

### 5.295.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Iterator.

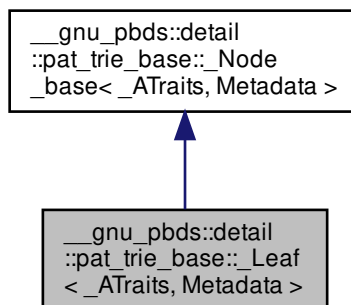
Definition at line 713 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 5.296 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf< \_ATraits, Metadata > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf< \_ATraits, Metadata >:



## Public Types

- `typedef _Alloc::template rebind< _ATraits > __rebind_at`
- `typedef _Alloc::template rebind< \_Node\_base > __rebind_n`
- `typedef _ATraits::const_iterator a_const_iterator`
- `typedef __rebind_at::other::const_pointer a_const_pointer`
- `typedef _ATraits access_traits`
- `typedef _Alloc allocator_type`
- `typedef \_Node\_base< _ATraits, Metadata > base_type`
- `typedef type_traits::const_reference const_reference`
- `typedef __rebind_n::other::pointer node_pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

## Public Member Functions

- `_Leaf (const_reference other)`
- `reference value ()`
- `const_reference value () const`

## Public Attributes

- `node_pointer m_p_parent`
- `const node\_type m_type`

## 5.296.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >
```

Leaf node for PATRICIA tree.

Definition at line 162 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

5.297 `__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >` Struct Template Reference

## Public Types

- `typedef _Alloc::template rebind< Metadata > __rebind_m`
- `typedef _Alloc allocator_type`
- `typedef __rebind_m::other::const_reference const_reference`
- `typedef Metadata metadata_type`

## Public Member Functions

- const\_reference **get\_metadata** () const

## Public Attributes

- metadata\_type **m\_metadata**

### 5.297.1 Detailed Description

```
template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
```

Metadata base primary template.

Definition at line 67 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 5.298 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Metadata< null\_type, \_Alloc > Struct Template Reference

## Public Types

- typedef \_Alloc **allocator\_type**
- typedef [null\\_type](#) **metadata\_type**

### 5.298.1 Detailed Description

```
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >
```

Specialization for null metadata.

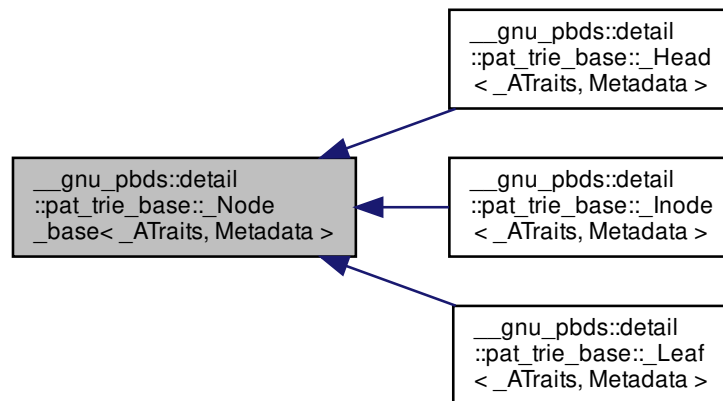
Definition at line 83 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 5.299 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base&lt; \_ATraits, Metadata &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base< \_ATraits, Metadata >:



## Public Types

- typedef \_\_Alloc::template rebind< \_ATraits > \_\_rebind\_at
- typedef \_\_Alloc::template rebind< [\\_Node\\_base](#) > \_\_rebind\_n
- typedef \_ATraits::const\_iterator **a\_const\_iterator**
- typedef \_\_rebind\_at::other::const\_pointer **a\_const\_pointer**
- typedef \_ATraits **access\_traits**
- typedef \_\_Alloc **allocator\_type**
- typedef \_\_rebind\_n::other::pointer **node\_pointer**
- typedef \_ATraits::type\_traits **type\_traits**

## Public Member Functions

- **\_Node\_base** ([node\\_type](#) type)

## Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**



### 5.299.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >
```

Node base.

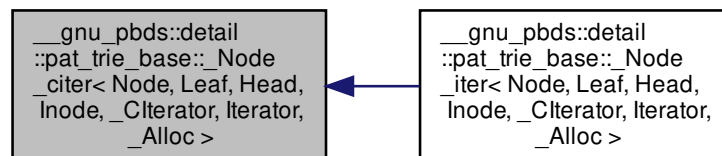
Definition at line 92 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 5.300 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >:



### Public Types

- typedef `_Alloc::template rebind< metadata\_type > __rebind_m`
- typedef `__rebind_m::other __rebind_ma`
- typedef `value_type const reference`
- typedef `trivial_iterator_difference_type difference_type`
- typedef `trivial_iterator_tag iterator_category`
- typedef `__rebind_ma::const_reference metadata_const_reference`
- typedef `Node::metadata_type metadata\_type`
- typedef `value_type reference`
- typedef `_Alloc::size_type size_type`
- typedef `_Clterator value_type`

## Public Member Functions

- `_Node_citer` (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- `_Node_citer get_child` (size\_type i) const
- metadata\_const\_reference `get_metadata` () const
- size\_type `num_children` () const
- bool `operator!=` (const `_Node_citer` &other) const
- const\_reference `operator*` () const
- bool `operator==` (const `_Node_citer` &other) const
- `std::pair< a_const_iterator, a_const_iterator >` `valid_prefix` () const

## Public Attributes

- node\_pointer `m_p_nd`
- a\_const\_pointer `m_p_traits`

## Protected Types

- typedef `_Alloc::template rebind< Inode >` `__rebind_in`
- typedef `_Alloc::template rebind< Leaf >` `__rebind_l`
- typedef `_Alloc::template rebind< Node >` `__rebind_n`
- typedef `Node::a_const_iterator` `a_const_iterator`
- typedef `Node::a_const_pointer` `a_const_pointer`
- typedef `__rebind_in::other::const_pointer` `inode_const_pointer`
- typedef `__rebind_in::other::pointer` `inode_pointer`
- typedef `__rebind_l::other::const_pointer` `leaf_const_pointer`
- typedef `__rebind_l::other::pointer` `leaf_pointer`
- typedef `__rebind_n::other::pointer` `node_pointer`

### 5.300.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _↵
Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
```

Node const iterator.

Definition at line 814 of file `pat_trie_base.hpp`.

### 5.300.2 Member Typedef Documentation

### 5.300.2.1 \_\_rebind\_m

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::__rebind_m
```

Const metadata reference type.

Definition at line 869 of file pat\_trie\_base.hpp.

### 5.300.2.2 metadata\_type

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

Definition at line 866 of file pat\_trie\_base.hpp.

## 5.300.3 Member Function Documentation

### 5.300.3.1 get\_child()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
__Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
    size_type i ) const [inline]
```

Returns a \_\_const node \_\_iterator to the corresponding node's i-th child.

Definition at line 911 of file pat\_trie\_base.hpp.

### 5.300.3.2 get\_metadata()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata ( ) const [inline]
```

Metadata access.

Definition at line 894 of file pat\_trie\_base.hpp.

#### 5.300.3.3 `num_children()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children ( ) const [inline]
```

Returns the number of children in the corresponding node.

Definition at line 899 of file `pat_trie_base.hpp`.

#### 5.300.3.4 `operator!=( )`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!=(
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other )
const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 927 of file `pat_trie_base.hpp`.

#### 5.300.3.5 `operator*()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _C
Iterator, Iterator, _Alloc >::operator* ( ) const [inline]
```

Const access; returns the `__const` iterator\* associated with the current leaf.

Definition at line 886 of file `pat_trie_base.hpp`.

#### 5.300.3.6 `operator==( )`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator==(
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other )
const [inline]
```

Compares content to a different iterator object.

Definition at line 922 of file `pat_trie_base.hpp`.

## 5.300.3.7 valid\_prefix()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix ( ) const [inline]
```

Subtree valid prefix.

Definition at line 880 of file pat\_trie\_base.hpp.

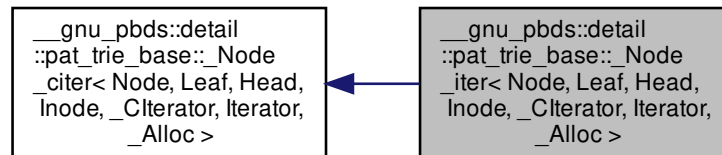
References `std::make_pair()`.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 5.301 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



#### Public Types

- typedef `_Alloc::template rebind< metadata_type > __rebind_m`
- typedef `__rebind_m::other __rebind_ma`
- typedef `value_type const_reference`
- typedef `trivial_iterator_difference_type difference_type`
- typedef `trivial_iterator_tag iterator_category`
- typedef `__rebind_ma::const_reference metadata_const_reference`
- typedef `Node::metadata_type metadata_type`
- typedef `value_type reference`
- typedef `base_type::size_type size_type`
- typedef `Iterator value_type`

## Public Member Functions

- `_Node_iter` (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- `_Node_iter get_child` (size\_type i) const
- metadata\_const\_reference `get_metadata` () const
- size\_type `num_children` () const
- bool `operator!=` (const `_Node_citer` &other) const
- reference `operator*` () const
- bool `operator==` (const `_Node_citer` &other) const
- `std::pair< a_const_iterator, a_const_iterator > valid_prefix` () const

## Public Attributes

- node\_pointer `m_p_nd`
- a\_const\_pointer `m_p_traits`

## Protected Types

- typedef `_Alloc::template rebind< Inode > __rebind_in`
- typedef `_Alloc::template rebind< Leaf > __rebind_l`
- typedef `Node::a_const_iterator a_const_iterator`
- typedef `__rebind_in::other::const_pointer inode_const_pointer`
- typedef `__rebind_l::other::const_pointer leaf_const_pointer`
- typedef `__rebind_l::other::pointer leaf_pointer`

### 5.301.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _↵
Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 943 of file `pat_trie_base.hpp`.

### 5.301.2 Member Typedef Documentation

#### 5.301.2.1 `__rebind_m`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::__rebind_m [inherited]
```

Const metadata reference type.

Definition at line 869 of file `pat_trie_base.hpp`.

### 5.301.2.2 metadata\_type

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]
```

Metadata type.

Definition at line 866 of file pat\_trie\_base.hpp.

### 5.301.3 Member Function Documentation

#### 5.301.3.1 get\_child()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
__Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
    size_type i ) const [inline]
```

Returns a node \_\_iterator to the corresponding node's i-th child.

Definition at line 976 of file pat\_trie\_base.hpp.

#### 5.301.3.2 get\_metadata()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata ( ) const [inline], [inherited]
```

Metadata access.

Definition at line 894 of file pat\_trie\_base.hpp.

#### 5.301.3.3 num\_children()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children ( ) const [inline], [inherited]
```

Returns the number of children in the corresponding node.

Definition at line 899 of file pat\_trie\_base.hpp.

#### 5.301.3.4 `operator!=(())`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!=(
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other )
const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 927 of file `pat_trie_base.hpp`.

#### 5.301.3.5 `operator*()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator* ( ) const [inline]
```

Access; returns the iterator\* associated with the current leaf.

Definition at line 968 of file `pat_trie_base.hpp`.

#### 5.301.3.6 `operator==(())`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator==(
    const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other )
const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 922 of file `pat_trie_base.hpp`.

#### 5.301.3.7 `valid_prefix()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix ( ) const [inline], [inherited]
```

Subtree valid prefix.

Definition at line 880 of file `pat_trie_base.hpp`.

References `std::make_pair()`.

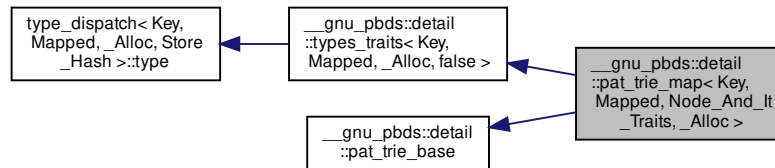
The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)



### 5.302 `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`:



#### Public Types

- typedef traits\_type::access\_traits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef `std::pair`< size\_type, size\_type > **comp\_hash**
- typedef point\_const\_iterator **const\_iterator**
- typedef traits\_base::const\_pointer **const\_pointer**
- typedef traits\_base::const\_reference **const\_reference**
- typedef traits\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef `pat_trie_tag` **container\_category**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef point\_iterator **iterator**
- typedef traits\_base::key\_const\_pointer **key\_const\_pointer**
- typedef traits\_base::key\_const\_reference **key\_const\_reference**
- typedef traits\_base::key\_pointer **key\_pointer**
- typedef traits\_base::key\_reference **key\_reference**
- typedef traits\_base::key\_type **key\_type**
- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef traits\_type::node\_const\_iterator **node\_const\_iterator**
- typedef traits\_type::node\_iterator **node\_iterator**
- enum `node_type` { **i\_node**, **leaf\_node**, **head\_node** }
- typedef traits\_type::node\_update **node\_update**
- typedef traits\_type::const\_iterator **point\_const\_iterator**
- typedef traits\_type::iterator **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef traits\_base::reference **reference**
- typedef traits\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef traits\_base::value\_type **value\_type**

## Public Member Functions

- **pat\_trie\_map** (const access\_traits &)
- **pat\_trie\_map** (const [pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- const\_iterator **erase** (const\_iterator)
- iterator **erase** (iterator)
- const\_reverse\_iterator **erase** (const\_reverse\_iterator)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- access\_traits & **get\_access\_traits** ()
- const access\_traits & **get\_access\_traits** () const
- node\_update & **get\_node\_update** ()
- const node\_update & **get\_node\_update** () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference)
- void **join** ([pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator [node\\_begin](#) () const
- node\_iterator [node\\_begin](#) ()
- node\_const\_iterator [node\\_end](#) () const
- node\_iterator [node\\_end](#) ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Member Functions

- `template<typename It >`  
`void copy_from_range (It, It)`
- `node_pointer recursive_copy_node (node_const_pointer)`
- `void value_swap (pat\_trie\_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)`

#### 5.302.1 Detailed Description

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >
```

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptsset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Definition at line 101 of file `pat_trie_.hpp`.

#### 5.302.2 Member Enumeration Documentation

##### 5.302.2.1 node\_type

```
enum \_\_gnu\_pbds::detail::pat\_trie\_base::node\_type [inherited]
```

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

Definition at line 58 of file `pat_trie_base.hpp`.

#### 5.302.3 Member Function Documentation

##### 5.302.3.1 node\_begin() [1/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat\_trie\_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator \_\_gnu\_pbds::detail::pat\_trie\_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ( ) const [inline]
```

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 101 of file `pat_trie_.hpp`.

### 5.302.3.2 `node_begin()` [2/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ( ) [inline]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 107 of file `pat_trie_.hpp`.

### 5.302.3.3 `node_end()` [1/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ( ) const [inline]
```

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 113 of file `pat_trie_.hpp`.

### 5.302.3.4 `node_end()` [2/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ( ) [inline]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 119 of file `pat_trie_.hpp`.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_.hpp](#)

## 5.303 `__gnu_pbds::detail::probe_fn_base<_Alloc>` Class Template Reference

### 5.303.1 Detailed Description

```
template<typename _Alloc>
class __gnu_pbds::detail::probe_fn_base<_Alloc>
```

Probe functor base.

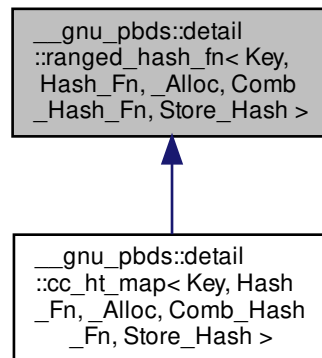
Definition at line 52 of file `probe_fn_base.hpp`.

The documentation for this class was generated from the following file:

- [probe\\_fn\\_base.hpp](#)

### 5.304 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`:



#### 5.304.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >
```

Primary template.

Definition at line 55 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 5.305 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

#### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

## Protected Member Functions

- `ranged_hash_fn` (size\_type)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void `notify_resized` (size\_type)
- size\_type `operator()` (key\_const\_reference) const
- void `swap` (`ranged_hash_fn`< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, false > &)

### 5.305.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
```

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 5.306 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

## Protected Types

- typedef `Comb_Hash_Fn` `comb_hash_fn_base`
- typedef `std::pair`< size\_type, size\_type > `comp_hash`
- typedef `Hash_Fn` `hash_fn_base`
- typedef `_Alloc::template rebind`< Key >::other `key_allocator`
- typedef `key_allocator::const_reference` `key_const_reference`
- typedef `_Alloc::size_type` `size_type`

## Protected Member Functions

- `ranged_hash_fn` (size\_type)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void `notify_resized` (size\_type)
- `comp_hash operator()` (key\_const\_reference) const
- `comp_hash operator()` (key\_const\_reference, size\_type) const
- void `swap` (`ranged_hash_fn`< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, true > &)

### 5.306.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
```

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

Definition at line 153 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 5.307 \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null\_type, \_Alloc, Comb\_Hash\_Fn, false > Class Template Reference

Inherits `Comb_Hash_Fn`.

#### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

#### Protected Member Functions

- **ranged\_hash\_fn** (`size_type`)
- **ranged\_hash\_fn** (`size_type`, `const Comb_Hash_Fn &`)
- **ranged\_hash\_fn** (`size_type`, `const null_type &`, `const Comb_Hash_Fn &`)
- void **swap** ([ranged\\_hash\\_fn](#)< `Key`, `null_type`, `_Alloc`, `Comb_Hash_Fn`, `false` > &)

### 5.307.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
```

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

Definition at line 255 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 5.308 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Comb_Hash_Fn`.

### Protected Types

- typedef `Comb_Hash_Fn` **`comb_hash_fn_base`**
- typedef `_Alloc::size_type` **`size_type`**

### Protected Member Functions

- **`ranged_hash_fn`** (`size_type`)
- **`ranged_hash_fn`** (`size_type`, `const Comb_Hash_Fn &`)
- **`ranged_hash_fn`** (`size_type`, `const null_type &`, `const Comb_Hash_Fn &`)
- void **`swap`** (`ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > &`)

### 5.308.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
```

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

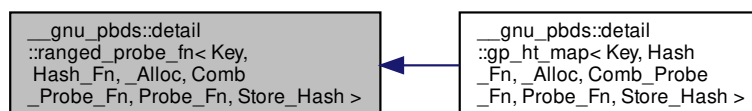
Definition at line 312 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 5.309 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`:





### 5.309.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >
```

Primary template.

Definition at line 55 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

### 5.310 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

#### Protected Types

- typedef `Comb_Probe_Fn` **comb\_probe\_fn\_base**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `_Alloc::template rebind< Key >::other` **key\_allocator**
- typedef `key_allocator::const_reference` **key\_const\_reference**
- typedef `Probe_Fn` **probe\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

#### Protected Member Functions

- **ranged\_probe\_fn** (`size_type`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`, `const Probe_Fn &`)
- void **notify\_resized** (`size_type`)
- `size_type` **operator()** (`key_const_reference`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`, `size_type`) `const`
- void **swap** ([ranged\\_probe\\_fn](#)< `Key`, `Hash_Fn`, `_Alloc`, `Comb_Probe_Fn`, `Probe_Fn`, `false` > &)

### 5.310.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

Specialization 1 The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 5.311 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

### Protected Types

- typedef `Comb_Probe_Fn` **`comb_probe_fn_base`**
- typedef `std::pair< size_type, size_type >` **`comp_hash`**
- typedef `Hash_Fn` **`hash_fn_base`**
- typedef `_Alloc::template rebind< Key >::other` **`key_allocator`**
- typedef `key_allocator::const_reference` **`key_const_reference`**
- typedef `Probe_Fn` **`probe_fn_base`**
- typedef `_Alloc::size_type` **`size_type`**

### Protected Member Functions

- **`ranged_probe_fn`** (`size_type`)
- **`ranged_probe_fn`** (`size_type`, `const Hash_Fn &`)
- **`ranged_probe_fn`** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`)
- **`ranged_probe_fn`** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`, `const Probe_Fn &`)
- **`notify_resized`** (`size_type`)
- **`comp_hash operator()`** (`key_const_reference`) `const`
- `size_type` **`operator()`** (`key_const_reference`, `size_type`, `size_type`) `const`
- `size_type` **`operator()`** (`key_const_reference`, `size_type`) `const`
- **`swap`** (`ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > &`)

#### 5.311.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
```

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 176 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 5.312 `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >` Class Template Reference

Inherits `Comb_Probe_Fn`.

### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef \_Alloc::template rebind< Key >::other **key\_allocator**
- typedef key\_allocator::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_probe\_fn** (size\_type size)
- **ranged\_probe\_fn** (size\_type, const Comb\_Probe\_Fn &r\_comb\_probe\_fn)
- **ranged\_probe\_fn** (size\_type, const [null\\_type](#) &, const Comb\_Probe\_Fn &r\_comb\_probe\_fn, const [null\\_type](#) &)
- void **swap** ([ranged\\_probe\\_fn](#) &other)

#### 5.312.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
```

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

Definition at line 296 of file [ranged\\_probe\\_fn.hpp](#).

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

#### 5.313 \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > Class Template Reference

Inherits [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#).

### Public Types

- typedef \_Alloc **allocator\_type**
- typedef Cmp\_Fn **cmp\_fn**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef base\_type::const\_pointer **const\_pointer**
- typedef base\_type::const\_reference **const\_reference**
- typedef base\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef [rb\\_tree\\_tag](#) **container\_category**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef base\_type::key\_const\_pointer **key\_const\_pointer**

- `typedef base_type::key_const_reference` **key\_const\_reference**
- `typedef base_type::key_pointer` **key\_pointer**
- `typedef base_type::key_reference` **key\_reference**
- `typedef base_type::key_type` **key\_type**
- `typedef base_type::mapped_const_pointer` **mapped\_const\_pointer**
- `typedef base_type::mapped_const_reference` **mapped\_const\_reference**
- `typedef base_type::mapped_pointer` **mapped\_pointer**
- `typedef base_type::mapped_reference` **mapped\_reference**
- `typedef base_type::mapped_type` **mapped\_type**
- `typedef __nothrowcopy::indicator` **no\_throw\_indicator**
- `typedef traits_type::node_const_iterator` **node\_const\_iterator**
- `typedef traits_type::node_iterator` **node\_iterator**
- `typedef base_type::node_update` **node\_update**
- `typedef base_type::const_iterator` **point\_const\_iterator**
- `typedef base_type::point_iterator` **point\_iterator**
- `typedef base_type::pointer` **pointer**
- `typedef base_type::reference` **reference**
- `typedef base_type::reverse_iterator` **reverse\_iterator**
- `typedef _Alloc::size_type` **size\_type**
- `typedef integral_constant< int, Store_Hash >` **store\_extra**
- `typedef base_type::value_type` **value\_type**

#### Public Member Functions

- **rb\_tree\_map** (const Cmp\_Fn &)
- **rb\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **rb\_tree\_map** (const [rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- `template<typename It >`  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- iterator **erase** (iterator)
- reverse\_iterator **erase** (reverse\_iterator)
- `template<typename Pred >`  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- `std::pair< point_iterator, bool >` **insert** (const\_reference)
- void **join** ([rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator **node\_begin** () const

- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_end** () const
- node\_iterator **node\_end** ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

#### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### Protected Types

- typedef node\_allocator::value\_type **node**
- typedef \_Alloc::template rebind< typename traits\_type::node >::other **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

#### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\* )
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\* )
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

#### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

#### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

#### Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

#### 5.313.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

Definition at line 84 of file `rb_tree_.hpp`.

#### 5.313.2 Member Function Documentation

##### 5.313.2.1 `node_begin()` [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↔
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↔
begin ( ) const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to the node at the root of the tree.

Definition at line 109 of file `bin_search_tree_.hpp`.

#### 5.313.2.2 `node_begin()` [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
( ) [inline], [inherited]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 117 of file `bin_search_tree_.hpp`.

#### 5.313.2.3 `node_end()` [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
end ( ) const [inline], [inherited]
```

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 125 of file `bin_search_tree_.hpp`.

#### 5.313.2.4 `node_end()` [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end (
) [inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following file:

- [rb\\_tree\\_.hpp](#)

### 5.314 `__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

#### Public Types

- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` **metadata\_const\_reference**
- typedef `_Alloc::template rebind< metadata_type >::other::reference` **metadata\_reference**
- typedef `Metadata` **metadata\_type**
- typedef `_Alloc::template rebind< rb\_tree\_node\_< Value\_Type, Metadata, \_Alloc > >::other::pointer` **node\_↵  
pointer**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `metadata_const_reference` **get\_metadata** () const
- `metadata_reference` **get\_metadata** ()
- `bool` **special** () const

## Public Attributes

- `metadata_type` **m\_metadata**
- `node_pointer` **m\_p\_left**
- `node_pointer` **m\_p\_parent**
- `node_pointer` **m\_p\_right**
- `bool` **m\_red**
- `value_type` **m\_value**

## 5.314.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for Red-Black trees.

Definition at line 52 of file `rb_tree_map_/node.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/node.hpp](#)

5.315 `__gnu_pbds::detail::rc< _Node, _Alloc >` Class Template Reference

## Public Types

- `typedef entry_const_pointer` **const\_iterator**
- `typedef node_pointer` **entry**

## Public Member Functions

- `rc` (const [rc](#) &)
- `const const_iterator` **begin** () const
- `void` **clear** ()
- `bool` **empty** () const
- `const const_iterator` **end** () const
- `void` **pop** ()
- `void` **push** (entry)
- `size_type` **size** () const
- `void` **swap** ([rc](#) &)
- `node_pointer` **top** () const



### 5.315.1 Detailed Description

```
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >
```

Redundant binary counter.

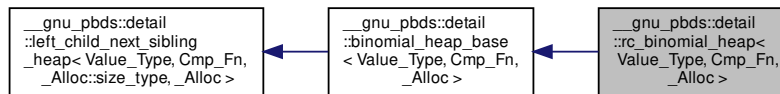
Definition at line 50 of file rc.hpp.

The documentation for this class was generated from the following file:

- [rc.hpp](#)

### 5.316 \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc >:



#### Public Types

- typedef base\_type::allocator\_type **allocator\_type**
- typedef base\_type::cmp\_fn **cmp\_fn**
- typedef [base\\_type::const\\_iterator](#) **const\_iterator**
- typedef base\_type::const\_pointer **const\_pointer**
- typedef base\_type::const\_reference **const\_reference**
- typedef \_\_Alloc::difference\_type **difference\_type**
- typedef [base\\_type::iterator](#) **iterator**
- typedef [base\\_type::point\\_const\\_iterator](#) **point\_const\_iterator**
- typedef [base\\_type::point\\_iterator](#) **point\_iterator**
- typedef base\_type::pointer **pointer**
- typedef base\_type::reference **reference**
- typedef \_\_Alloc::size\_type **size\_type**
- typedef Value\_Type **value\_type**

## Public Member Functions

- `rc_binomial_heap` (const Cmp\_Fn &)
- `rc_binomial_heap` (const `rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- `iterator begin` ()
- `const_iterator begin` () const
- void `clear` ()
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- void `erase` (`point_iterator`)
- template<typename Pred >  
size\_type `erase_if` (Pred)
- Cmp\_Fn & `get_cmp_fn` ()
- const Cmp\_Fn & `get_cmp_fn` () const
- void `join` (`rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `join` (`binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type `max_size` () const
- void `modify` (`point_iterator`, const\_reference)
- void `pop` ()
- `point_iterator push` (const\_reference)
- size\_type `size` () const
- template<typename Pred >  
void `split` (Pred, `rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- template<typename Pred >  
void `split` (Pred, `binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `swap` (`rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `swap` (`left_child_next_sibling_heap`< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference `top` () const

## Protected Types

- typedef base\_type::node `node`
- typedef \_Alloc::template rebind< `left_child_next_sibling_heap_node`< Value\_Type, \_Alloc::size\_type, \_Alloc >::other `node_allocator`
- typedef \_Alloc::size\_type `node_metadata`
- typedef `std::pair`< node\_pointer, node\_pointer > `node_pointer_pair`

## Protected Member Functions

- void `actual_erase_node` (node\_pointer)
- void `bubble_to_top` (node\_pointer)
- void `clear_imp` (node\_pointer)
- template<typename It >  
void `copy_from_range` (It, It)
- void `find_max` ()
- node\_pointer `get_new_node_for_insert` (const\_reference)
- node\_pointer `prune` (Pred)
- void `swap` (`binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `swap_with_parent` (node\_pointer, node\_pointer)
- void `to_linked_list` ()
- void `value_swap` (`left_child_next_sibling_heap` &)

#### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

#### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.316.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Redundant-counter binomial heap.

Definition at line 66 of file rc\_binomial\_heap\_.hpp.

The documentation for this class was generated from the following file:

- [rc\\_binomial\\_heap\\_.hpp](#)

#### 5.317 \_\_gnu\_pbds::detail::resize\_policy<\_Tp> Class Template Reference

##### Public Types

- typedef \_Tp **size\_type**

##### Public Member Functions

- **resize\_policy** (const [resize\\_policy](#) &other)
- size\_type **get\_new\_size\_for\_arbitrary** (size\_type) const
- size\_type **get\_new\_size\_for\_grow** () const
- size\_type **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (size\_type) const
- void **notify\_arbitrary** (size\_type)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- bool **resize\_needed\_for\_grow** (size\_type) const
- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- void **swap** ([resize\\_policy](#)<\_Tp> &)

## Static Public Attributes

- static const `_Tp` **min\_size**

## 5.317.1 Detailed Description

```
template<typename _Tp>
class __gnu_pbds::detail::resize_policy< _Tp >
```

Resize policy for binary heap.

Definition at line 52 of file `resize_policy.hpp`.

The documentation for this class was generated from the following file:

- [resize\\_policy.hpp](#)

## 5.318 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `splay_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**

- typedef base\_type::node\_update **node\_update**
- typedef base\_type::const\_iterator **point\_const\_iterator**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef base\_type::pointer **pointer**
- typedef base\_type::reference **reference**
- typedef base\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef base\_type::value\_type **value\_type**

#### Public Member Functions

- **splay\_tree\_map** (const Cmp\_Fn &)
- **splay\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **splay\_tree\_map** (const [splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It >  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- iterator **erase** (iterator it)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **initialize** ()
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_value)
- void **join** ([splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_end** () const
- node\_iterator **node\_end** ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, [splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Protected Types

- typedef node\_allocator::value\_type **node**
- typedef \_Alloc::template rebind< typename traits\_type::node >::other **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

## Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

## Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

## Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

### 5.318.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Splay tree.

Definition at line 107 of file splay\_tree\_.hpp.

### 5.318.2 Member Function Documentation

#### 5.318.2.1 node\_begin() [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
begin ( ) const [inline], [inherited]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

Definition at line 109 of file bin\_search\_tree\_.hpp.

#### 5.318.2.2 node\_begin() [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
( ) [inline], [inherited]
```

Returns a node\_iterator corresponding to the node at the root of the tree.

Definition at line 117 of file bin\_search\_tree\_.hpp.

## 5.319 `__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference#513

### 5.318.2.3 `node_end()` [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
end ( ) const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 125 of file `bin_search_tree_.hpp`.

### 5.318.2.4 `node_end()` [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end (
) [inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following file:

- [splay\\_tree\\_.hpp](#)

## 5.319 `__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

### Public Types

- `typedef _Alloc::template rebind< metadata_type >::other::const_reference metadata_const_reference`
- `typedef _Alloc::template rebind< metadata_type >::other::reference metadata_reference`
- `typedef Metadata metadata_type`
- `typedef _Alloc::template rebind< splay\_tree\_node\_< Value\_Type, Metadata, \_Alloc > >::other::pointer node_↵_pointer`
- `typedef Value_Type value_type`

### Public Member Functions

- `metadata_const_reference get_metadata () const`
- `metadata_reference get_metadata ()`
- `bool special () const`



## Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_special**
- value\_type **m\_value**

### 5.319.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for splay tree.

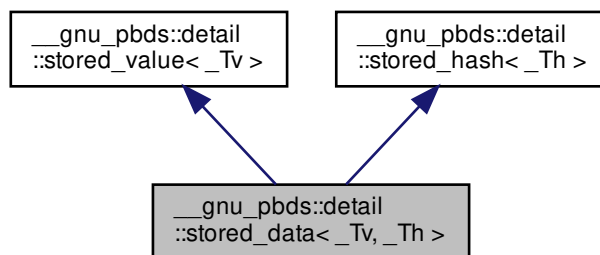
Definition at line 50 of file splay\_tree\_/node.hpp.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/node.hpp](#)

### 5.320 \_\_gnu\_pbds::detail::stored\_data<\_Tv,\_Th> Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::stored\_data<\_Tv,\_Th>:



## Public Types

- typedef `_Th` **hash\_type**
- typedef `_Tv` **value\_type**

#### Public Attributes

- hash\_type **m\_hash**
- value\_type **m\_value**

#### 5.320.1 Detailed Description

```
template<typename _Tv, typename _Th>
struct __gnu_pbds::detail::stored_data<_Tv, _Th>
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.

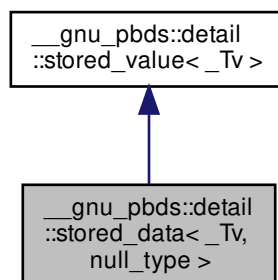
Definition at line 95 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 5.321 `__gnu_pbds::detail::stored_data<_Tv, null_type>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, null_type>`:



#### Public Types

- typedef `_Tv` **value\_type**

#### Public Attributes

- value\_type **m\_value**

### 5.321.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_data<_Tv, null_type >
```

Specialization for representation of stored data of just value type.

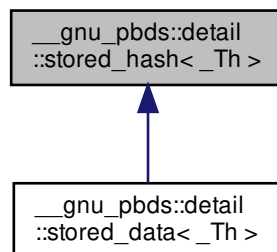
Definition at line 101 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

### 5.322 \_\_gnu\_pbds::detail::stored\_hash<\_Th > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::stored\_hash<\_Th >:



#### Public Types

- typedef `_Th` **hash\_type**

#### Public Attributes

- `hash_type` **m\_hash**

## 5.322.1 Detailed Description

```
template<typename _Th>
struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

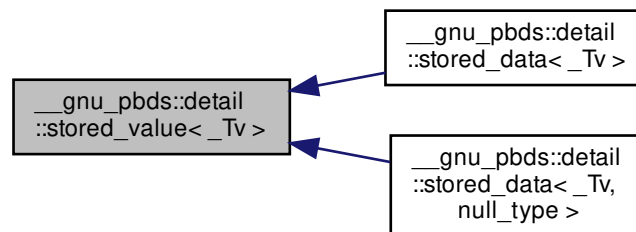
Definition at line 86 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 5.323 \_\_gnu\_pbds::detail::stored\_value&lt;\_Tv&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::stored\_value<\_Tv>:



## Public Types

- typedef `_Tv` **value\_type**

## Public Attributes

- value\_type **m\_value**

## 5.323.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

Definition at line 78 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

### 5.324 `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >` Struct Template Reference

Inherits `_ATraits`.

#### Public Types

- typedef `_ATraits` **base\_type**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `type_traits::const_reference` **const\_reference**
- typedef `type_traits::key_const_reference` **key\_const\_reference**
- typedef `Type_Traits` **type\_traits**

#### Public Member Functions

- **synth\_access\_traits** (const `base_type` &)
- bool **cmp\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **cmp\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=false) const
- bool **equal\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **equal\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=true) const

#### Static Public Member Functions

- static key\_const\_reference **extract\_key** (const\_reference)

#### 5.324.1 Detailed Description

```
template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >
```

Synthetic element access traits.

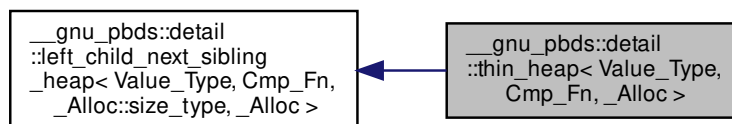
Definition at line 59 of file `synth_access_traits.hpp`.

The documentation for this struct was generated from the following file:

- [synth\\_access\\_traits.hpp](#)

### 5.325 `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`thin_heap< Value_Type, Cmp_Fn, _Alloc >` &)
- size\_type **max\_size** () const
- void **modify** (`point_iterator`, const\_reference)
- void **pop** ()
- `point_iterator` **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, `thin_heap< Value_Type, Cmp_Fn, _Alloc >` &)
- void **swap** (`left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc >` &)
- const\_reference **top** () const

## Protected Types

- typedef `base_type::node` **node**
- typedef `_Alloc::template rebind< left_child_next_sibling_heap_node_< Value_Type, _Alloc::size_type, _Alloc >::other` **node\_allocator**
- typedef `base_type::node_const_pointer` **node\_const\_pointer**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef `base_type::node_pointer` **node\_pointer**
- typedef `std::pair< node_pointer, node_pointer >` **node\_pointer\_pair**

### Protected Member Functions

- **thin\_heap** (const Cmp\_Fn &)
- **thin\_heap** (const [thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It >  
void **copy\_from\_range** (It, It)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([thin\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 5.325.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>  
class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >
```

Thin heap.

See Tarjan and Kaplan.

Definition at line 77 of file thin\_heap.hpp.

The documentation for this class was generated from the following file:

- [thin\\_heap.hpp](#)

5.326 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

## 5.326.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >
```

Tree metadata helper.

Definition at line 58 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

5.327 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

## Public Types

- typedef `Node_Update::metadata_type` **type**

## 5.327.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

5.328 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >` Struct Template Reference

## Public Types

- typedef `null_type` **type**



### 5.328.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 5.329 \_\_gnu\_pbds::detail::tree\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc > Struct Template Reference

### Public Types

- typedef [tree\\_metadata\\_helper](#)< \_\_node\_u, null\_update >::type **type**

### 5.329.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename
Cmp_Fn_, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

Definition at line 84 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 5.330 \_\_gnu\_pbds::detail::tree\_traits< Key, Data, Cmp\_Fn, Node\_Update, Tag, \_Alloc > Struct Template Reference

### 5.330.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_↵
Fn_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >
```

Tree traits class, primary template.

Definition at line 70 of file branch\_policy/traits.hpp.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 5.331 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` **node\_const\_iterator**
- typedef `ov_tree_node_it_< value_type, metadata_type, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`

### 5.331.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Tree traits.

Definition at line 61 of file `ov_tree_map_/traits.hpp`.

### 5.331.2 Member Typedef Documentation

#### 5.331.2.1 `node_const_iterator`

```
template<typename Key , typename Mapped , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

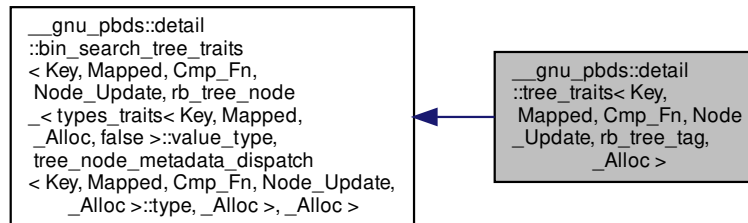
Definition at line 95 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

### 5.332 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



#### Public Types

- typedef `bin_search_tree_const_it< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

#### 5.332.1 Detailed Description

```

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

```

Specialization.

Definition at line 61 of file `rb_tree_map_traits.hpp`.

## 5.332.2 Member Typedef Documentation

### 5.332.2.1 `node_const_iterator`

```
typedef bin_search_tree_const_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↵
Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node< types_traits< Key, Mapped, _Alloc, false >↵
::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↵
Alloc > , _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

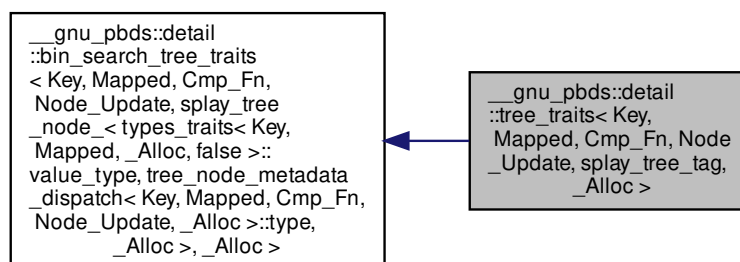
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

## 5.333 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



## Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, [\\_Alloc](#) > **const\_reverse\_iterator**
- typedef [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, [\\_Alloc](#), false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, [\\_Alloc](#) >::type, [\\_Alloc](#) > **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, [\\_Alloc](#), false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, [\\_Alloc](#) >::type, [\\_Alloc](#) >, [point\\_const\\_iterator](#), [point\\_iterator](#), [\\_Alloc](#) > **node\_const\_iterator**
- typedef [bin\\_search\\_tree\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, [\\_Alloc](#), false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, [\\_Alloc](#) >::type, [\\_Alloc](#) >, [point\\_const\\_iterator](#), [point\\_iterator](#), [\\_Alloc](#) > **node\_iterator**
- typedef Node\_Update < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, [\\_Alloc](#) > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#) < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, [\\_Alloc](#) > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, [\\_Alloc](#) > **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, [\\_Alloc](#) > **point\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, [\\_Alloc](#) > **reverse\_iterator**

## 5.333.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

Definition at line 61 of file [splay\\_tree\\_/traits.hpp](#).

## 5.333.2 Member Typedef Documentation

## 5.333.2.1 node\_const\_iterator

```
typedef bin\_search\_tree\_const\_node\_it < splay\_tree\_node < types\_traits < Key, Mapped, \_Alloc, false >::value_type, tree\_node\_metadata\_dispatch < Key, Mapped, Cmp_Fn, Node_Update, \_Alloc >::type, \_Alloc >, point\_const\_iterator, point\_iterator, \_Alloc > \_\_gnu\_pbds::detail::bin\_search\_tree\_traits < Key, Mapped, Cmp_Fn, Node_Update, splay\_tree\_node < types\_traits < Key, Mapped, \_Alloc, false >::value_type, tree\_node\_metadata\_dispatch < Key, Mapped, Cmp_Fn, Node_Update, \_Alloc >::type, \_Alloc >, \_Alloc >::node\_const\_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file [bin\\_search\\_tree\\_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

## 5.334 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_const_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`

### 5.334.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Specialization.

Definition at line 132 of file `ov_tree_map_/traits.hpp`.

### 5.334.2 Member Typedef Documentation

#### 5.334.2.1 `node_const_iterator`

```
template<typename Key , class Cmp_Fn , template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

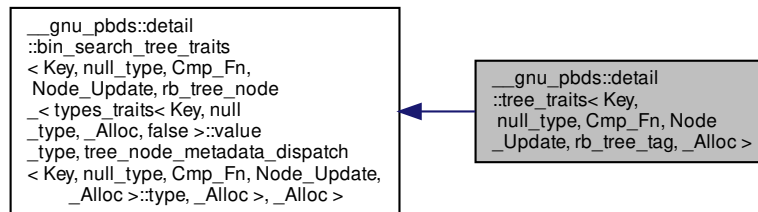
Definition at line 166 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

### 5.335 \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >:



#### Public Types

- typedef `bin_search_tree_const_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **const\_reverse\_iterator**
- typedef `rb_tree_node` < `types_traits` < Key, `null_type`, `_Alloc`, false >::value\_type, `tree_node_metadata_dispatch` < Key, `null_type`, `Cmp_Fn`, `Node_Update`, `_Alloc` >::type, `_Alloc` > **node**
- typedef `bin_search_tree_const_node_it` < `rb_tree_node` < `types_traits` < Key, `null_type`, `_Alloc`, false >::value\_type, `tree_node_metadata_dispatch` < Key, `null_type`, `Cmp_Fn`, `Node_Update`, `_Alloc` >::type, `_Alloc` >, `point_const_iterator`, `point_iterator`, `_Alloc` > **node\_const\_iterator**
- typedef `bin_search_tree_node_it` < `rb_tree_node` < `types_traits` < Key, `null_type`, `_Alloc`, false >::value\_type, `tree_node_metadata_dispatch` < Key, `null_type`, `Cmp_Fn`, `Node_Update`, `_Alloc` >::type, `_Alloc` >, `point_const_iterator`, `point_iterator`, `_Alloc` > **node\_iterator**
- typedef `Node_Update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > **node\_update**
- typedef `__gnu_pbds::null_node_update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > \* **null\_node\_update\_pointer**
- typedef `bin_search_tree_const_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point\_const\_iterator**
- typedef `bin_search_tree_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point\_iterator**
- typedef `bin_search_tree_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **reverse\_iterator**

#### 5.335.1 Detailed Description

```

template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

```

Specialization.

Definition at line 85 of file `rb_tree_map_/traits.hpp`.

### 5.335.2 Member Typedef Documentation

#### 5.335.2.1 `node_const_iterator`

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↵
::type, _Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type , Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type,
_Alloc > , _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

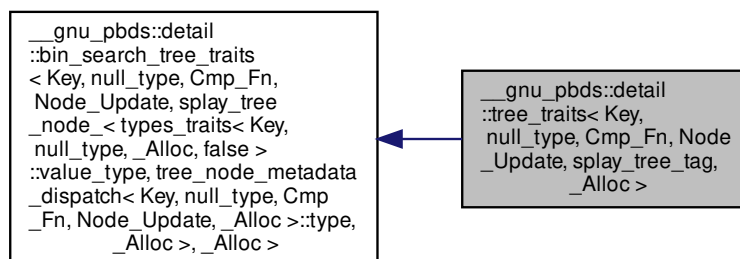
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

### 5.336 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:





## Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, [\\_Alloc](#) > **const\_reverse\_iterator**
- typedef [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, [null\\_type](#), [\\_Alloc](#), false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, [\\_Alloc](#) >::type, [\\_Alloc](#) > **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, [null\\_type](#), [\\_Alloc](#), false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, [\\_Alloc](#) >::type, [\\_Alloc](#) >, [point\\_const\\_iterator](#), [point\\_iterator](#), [\\_Alloc](#) > **node\_const\_iterator**
- typedef [bin\\_search\\_tree\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, [null\\_type](#), [\\_Alloc](#), false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, [\\_Alloc](#) >::type, [\\_Alloc](#) >, [point\\_const\\_iterator](#), [point\\_iterator](#), [\\_Alloc](#) > **node\_iterator**
- typedef Node\_Update < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, [\\_Alloc](#) > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#) < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, [\\_Alloc](#) > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, [\\_Alloc](#) > **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, [\\_Alloc](#) > **point\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename [\\_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, [\\_Alloc](#) > **reverse\_iterator**

## 5.336.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

Definition at line 81 of file [splay\\_tree\\_/traits.hpp](#).

## 5.336.2 Member Typedef Documentation

## 5.336.2.1 node\_const\_iterator

```
typedef bin\_search\_tree\_const\_node\_it < splay\_tree\_node < types\_traits < Key, null\_type, \_Alloc, false >::value_type, tree\_node\_metadata\_dispatch < Key, null\_type, Cmp_Fn, Node_Update, \_Alloc >::type, \_Alloc >, point\_const\_iterator, point\_iterator, \_Alloc > \_\_gnu\_pbds::detail::bin\_search\_tree\_traits < Key, null\_type, Cmp_Fn, Node_Update, splay\_tree\_node < types\_traits < Key, null\_type, \_Alloc, false >::value_type, tree\_node\_metadata\_dispatch < Key, null\_type, Cmp_Fn, Node_Update, \_Alloc >::type, \_Alloc >, \_Alloc >::node\_const\_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file [bin\\_search\\_tree\\_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

### 5.337 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, \_BTp > Struct Template Reference

#### 5.337.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

Definition at line 58 of file trie\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.338 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, false > Struct Template Reference

#### Public Types

- typedef Node\_Update::metadata\_type **type**

#### 5.338.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file trie\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.339 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, true > Struct Template Reference

#### Public Types

- typedef [null\\_type](#) **type**

### 5.339.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file trie\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.340 \_\_gnu\_pbds::detail::trie\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc > Struct Template Reference

#### Public Types

- typedef [trie\\_metadata\\_helper](#)< \_\_node\_u, null\_update >::type **type**

### 5.340.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Trie node metadata dispatch.

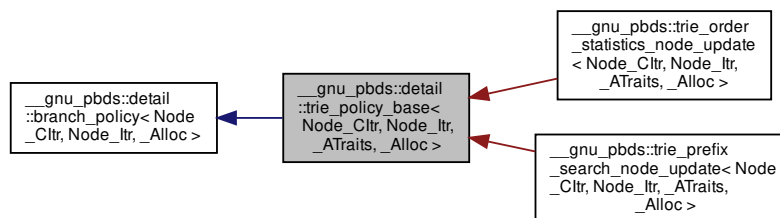
Definition at line 84 of file trie\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

### 5.341 \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >:



## Public Types

- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Protected Member Functions

- virtual `const_iterator` **end** () const =0
- virtual `iterator` **end** ()=0
- `it_type` **end\_iterator** () const
- virtual `const access_traits &` **get\_access\_traits** () const =0
- virtual `node_const_iterator` **node\_begin** () const =0
- virtual `node_iterator` **node\_begin** ()=0
- virtual `node_const_iterator` **node\_end** () const =0
- virtual `node_iterator` **node\_end** ()=0

## Static Protected Member Functions

- static `size_type` **common\_prefix\_len** (`node_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits &`)
- static `key_const_reference` **extract\_key** (`const_reference r_val`)
- static `iterator` **leftmost\_it** (`node_iterator`)
- static `bool` **less** (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits &`)
- static `iterator` **rightmost\_it** (`node_iterator`)

### 5.341.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

Base class for trie policies.

Definition at line 53 of file `trie_policy_base.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy\\_base.hpp](#)

## 5.342 \_\_gnu\_pbds::detail::trie\_traits< Key, Data, \_ATraits, Node\_Update, Tag, \_Alloc > Struct Template Reference

### 5.342.1 Detailed Description

```
template<typename Key, typename Data, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename _ATraits,
_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >
```

Trie traits class, primary template.

Definition at line 83 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 5.343 \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc > Struct Template Reference

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `base_type::_Iter< node, leaf, head, inode, true >` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `base_type::_Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > *` **null\_node\_update\_pointer**
- typedef `base_type::_Iter< node, leaf, head, inode, false >` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits >` **synth\_access\_traits**

#### 5.343.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CIttr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc_>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 62 of file `pat_trie_/traits.hpp`.

#### 5.343.2 Member Typedef Documentation

##### 5.343.2.1 `node_const_iterator`

```
template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc_ >
typedef base\_type::Node\_citer<node, leaf, head, inode, const\_iterator, iterator, \_Alloc> \_\_gnu\_pbds::detail::trie\_traits<
Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >::node\_const\_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 88 of file `pat_trie_/traits.hpp`.

##### 5.343.2.2 `node_update`

```
template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc_ >
typedef Node_Update<node\_const\_iterator, node\_iterator, _ATraits, _Alloc> \_\_gnu\_pbds::detail::trie\_traits<
Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >::node\_update
```

Type for node update.

Definition at line 93 of file `pat_trie_/traits.hpp`.

##### 5.343.2.3 `synth_access_traits`

```
template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc_ >
typedef \_\_gnu\_pbds::detail::synth\_access\_traits<type\_traits, false, access\_traits> \_\_gnu\_pbds::detail::trie\_traits<
Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >::synth\_access\_traits
```

Type for synthesized traits.

Definition at line 74 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 5.344 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > * null_node_update`↵  
**\_pointer**
- typedef `const_reverse_iterator` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits >` **synth\_access\_traits**

### 5.344.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _↵
_Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 109 of file `pat_trie_/traits.hpp`.

### 5.344.2 Member Typedef Documentation

#### 5.344.2.1 `node_const_iterator`

```
template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr,
typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tr
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 135 of file `pat_trie_/traits.hpp`.

5.344.2.2 `node_update`

```
template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr,
typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

Definition at line 140 of file `pat_trie_/traits.hpp`.

5.344.2.3 `synth_access_traits`

```
template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr,
typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

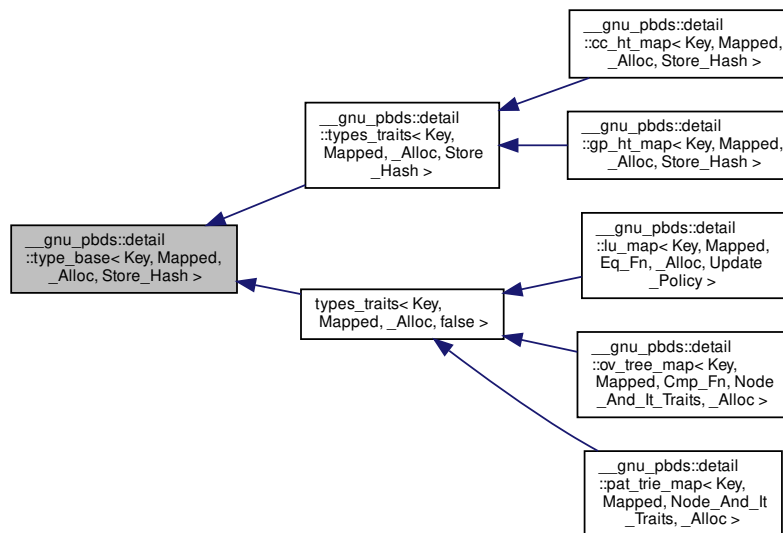
Definition at line 121 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

5.345 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`:





#### 5.345.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >
```

Primary template.

Definition at line 107 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 5.346 \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc, false > Struct Template Reference

##### Public Types

- typedef \_\_rebind\_va::const\_pointer **const\_pointer**
- typedef \_\_rebind\_va::const\_reference **const\_reference**
- typedef \_\_rebind\_ma::const\_pointer **mapped\_const\_pointer**
- typedef \_\_rebind\_ma::const\_reference **mapped\_const\_reference**
- typedef \_\_rebind\_ma::pointer **mapped\_pointer**
- typedef \_\_rebind\_ma::reference **mapped\_reference**
- typedef \_\_rebind\_ma::value\_type **mapped\_type**
- typedef \_\_rebind\_va::pointer **pointer**
- typedef \_\_rebind\_va::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef [stored\\_data](#)< value\_type, [null\\_type](#) > **stored\_data\_type**
- typedef \_\_rebind\_va::value\_type **value\_type**

#### 5.346.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >
```

Specialization of type\_base for the case where the hash value is not stored alongside each value.

Definition at line 114 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

5.347 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >` Struct Template Reference

## Public Types

- `typedef __rebind_va::const_pointer const_pointer`
- `typedef __rebind_va::const_reference const_reference`
- `typedef __rebind_ma::const_pointer mapped_const_pointer`
- `typedef __rebind_ma::const_reference mapped_const_reference`
- `typedef __rebind_ma::pointer mapped_pointer`
- `typedef __rebind_ma::reference mapped_reference`
- `typedef __rebind_ma::value_type mapped_type`
- `typedef __rebind_va::pointer pointer`
- `typedef __rebind_va::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef stored\_data< value_type, size_type > stored_data_type`
- `typedef __rebind_va::value_type value_type`

## 5.347.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >
```

Specialization of `type_base` for the case where the hash value is stored alongside each value.

Definition at line 147 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

5.348 `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >` Struct Template Reference

## Public Types

- `typedef __rebind_va::const_pointer const_pointer`
- `typedef __rebind_va::const_reference const_reference`
- `typedef __rebind_ma::const_pointer mapped_const_pointer`
- `typedef __rebind_ma::const_reference mapped_const_reference`
- `typedef __rebind_ma::pointer mapped_pointer`
- `typedef __rebind_ma::reference mapped_reference`
- `typedef __rebind_ma::value_type mapped_type`
- `typedef __rebind_va::pointer pointer`
- `typedef __rebind_va::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef stored\_data< value_type, null\_type > stored_data_type`
- `typedef Key value_type`

#### Static Public Attributes

- static [null\\_type](#) **s\_null\_type**

##### 5.348.1 Detailed Description

```
template<typename Key, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >
```

Specialization of type\_base for the case where the hash value is not stored alongside each value.

Definition at line 181 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 5.349 \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc, true > Struct Template Reference

##### Public Types

- typedef \_\_rebind\_va::const\_pointer **const\_pointer**
- typedef \_\_rebind\_va::const\_reference **const\_reference**
- typedef \_\_rebind\_ma::const\_pointer **mapped\_const\_pointer**
- typedef \_\_rebind\_ma::const\_reference **mapped\_const\_reference**
- typedef \_\_rebind\_ma::pointer **mapped\_pointer**
- typedef \_\_rebind\_ma::reference **mapped\_reference**
- typedef \_\_rebind\_ma::value\_type **mapped\_type**
- typedef \_\_rebind\_va::pointer **pointer**
- typedef \_\_rebind\_va::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef [stored\\_data](#)< value\_type, size\_type > **stored\_data\_type**
- typedef Key **value\_type**

#### Static Public Attributes

- static [null\\_type](#) **s\_null\_type**

##### 5.349.1 Detailed Description

```
template<typename Key, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >
```

Specialization of type\_base for the case where the hash value is stored alongside each value.

Definition at line 220 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

5.350 `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

## Public Types

- typedef `type_base< Key, Mapped, _Alloc, Store_Hash >` **type**

## 5.350.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >
```

Type base dispatch.

Definition at line 256 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- `types_traits.hpp`

5.351 `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`:



## Public Types

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `__rebind_a::const_pointer` **key\_const\_pointer**
- typedef `__rebind_a::const_reference` **key\_const\_reference**
- typedef `__rebind_a::pointer` **key\_pointer**
- typedef `__rebind_a::reference` **key\_reference**
- typedef `__rebind_a::value_type` **key\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### 5.351.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >
```

Traits for abstract types.

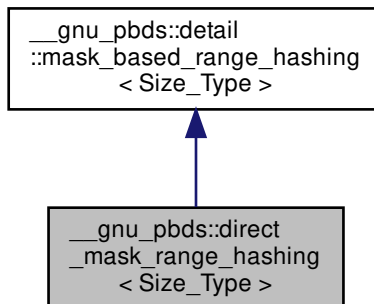
Definition at line 263 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 5.352 \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size\_Type > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size\_Type >:



### Public Types

- typedef Size\_Type **size\_type**

### Public Member Functions

- void **swap** ([direct\\_mask\\_range\\_hashing](#)< Size\_Type > &other)

## Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type **operator()** (size\_type hash) const
- size\_type **range\_hash** (size\_type hash) const
- void **swap** (mask\_based\_range\_hashing &other)

## 5.352.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

Definition at line 109 of file hash\_policy.hpp.

## 5.352.2 Member Function Documentation

5.352.2.1 `operator()`

```
template<typename Size_Type >
direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() (
    size_type hash ) const    [inline], [protected]
```

Transforms the `__hash` value hash into a ranged-hash value (using a bit-mask).

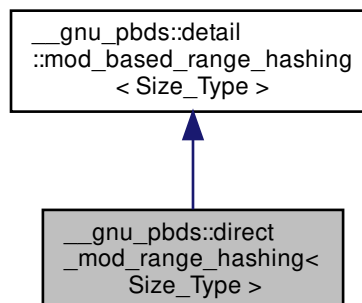
Definition at line 57 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

5.353 `__gnu_pbds::direct_mod_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mod_range_hashing< Size_Type >`:



## Public Types

- typedef Size\_Type **size\_type**

## Public Member Functions

- void **swap** ([direct\\_mod\\_range\\_hashing](#)< Size\_Type > &other)

## Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type [operator\(\)](#) (size\_type hash) const
- size\_type **range\_hash** (size\_type s) const
- void **swap** (mod\_based\_range\_hashing &other)

### 5.353.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >
```

A mod range-hashing class (uses the modulo function).

Definition at line 141 of file hash\_policy.hpp.

### 5.353.2 Member Function Documentation

#### 5.353.2.1 [operator\(\)](#)

```
template<typename Size_Type >
direct\_mod\_range\_hashing< Size_Type >::size_type \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size_Type
>::operator() (
    size_type hash ) const    [inline], [protected]
```

Transforms the `__hash` value hash into a ranged-hash value (using a modulo operation).

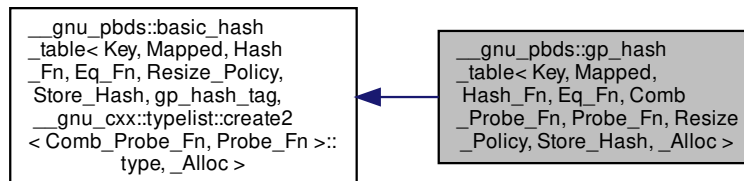
Definition at line 57 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

5.354 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Probe\_Fn **probe\_fn**
- typedef Resize\_Policy **resize\_policy**

#### Public Member Functions

- [gp\\_hash\\_table](#) ()
- [gp\\_hash\\_table](#) (const hash\_fn &h)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- **gp\_hash\_table** (const [gp\\_hash\\_table](#) &other)
- [gp\\_hash\\_table](#) & **operator=** (const [gp\\_hash\\_table](#) &other)
- void **swap** ([gp\\_hash\\_table](#) &other)



### 5.354.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn
= typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<↵
Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A general-probing hash-based associative container.

#### Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Probe_Fn</i>	Combining probe functor. If <i>Hash_Fn</i> is not <code>null_type</code> , then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies.
<i>Probe_Fn</i>	Probe functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If <i>Hash_Fn</i> is <code>null_type</code> , then the container will not compile if this value is true
<i>_Alloc</i>	Allocator type.

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

Definition at line 368 of file `assoc_container.hpp`.

### 5.354.2 Constructor & Destructor Documentation

#### 5.354.2.1 `gp_hash_table()` [1/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table ( ) [inline]
```

Default constructor.

Definition at line 382 of file `assoc_container.hpp`.

5.354.2.2 `gp_hash_table()` [2/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 386 of file `assoc_container.hpp`.

5.354.2.3 `gp_hash_table()` [3/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 393 of file `assoc_container.hpp`.

5.354.2.4 `gp_hash_table()` [4/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 401 of file `assoc_container.hpp`.

5.354.2.5 `gp_hash_table()` [5/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 410 of file `assoc_container.hpp`.

5.354.2.6 `gp_hash_table()` [6/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p,
    const resize_policy & rp ) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

Definition at line 422 of file `assoc_container.hpp`.

5.354.2.7 `gp_hash_table()` [7/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last ) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 430 of file `assoc_container.hpp`.

5.354.2.8 `gp_hash_table()` [8/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 438 of file `assoc_container.hpp`.

5.354.2.9 `gp_hash_table()` [9/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
```

```
const hash_fn & h,
const eq_fn & e ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 449 of file `assoc_container.hpp`.

#### 5.354.2.10 `gp_hash_table()` [10/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 461 of file `assoc_container.hpp`.

#### 5.354.2.11 `gp_hash_table()` [11/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 475 of file `assoc_container.hpp`.

5.354.2.12 `gp_hash_table()` [12/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
    It first,
    It last,
    const hash_fn & h,
    const eq_fn & e,
    const comb_probe_fn & cp,
    const probe_fn & p,
    const resize_policy & rp ) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_↵probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

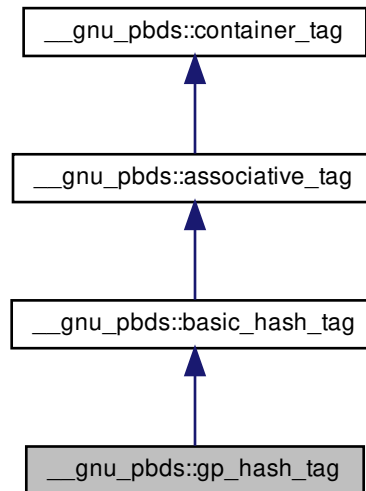
Definition at line 491 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.355 `__gnu_pbds::gp_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



#### 5.355.1 Detailed Description

General-probing hash.

Definition at line 144 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.356 `__gnu_pbds::hash_exponential_size_policy< Size_Type >` Class Template Reference

#### Public Types

- typedef `Size_Type` **size\_type**

#### Public Member Functions

- [hash\\_exponential\\_size\\_policy](#) (`size_type` start\_size=8, `size_type` grow\_factor=2)
- void **swap** ([hash\\_exponential\\_size\\_policy](#)< `Size_Type` > &other)

## Protected Member Functions

- `size_type get_nearest_larger_size (size_type size) const`
- `size_type get_nearest_smaller_size (size_type size) const`

### 5.356.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >
```

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).

Definition at line 413 of file `hash_policy.hpp`.

### 5.356.2 Constructor & Destructor Documentation

#### 5.356.2.1 `hash_exponential_size_policy()`

```
template<typename Size_Type >
__gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (
    size_type start_size = 8,
    size_type grow_factor = 2 )
```

Default constructor, or onstructor taking a `start_size`, or constructor taking a start size and `grow_factor`. The policy will use the sequence of sizes `start_size`, `start_size* grow_factor`, `start_size* grow_factor^2`, ...

Definition at line 44 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 5.357 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`:





### Public Types

- enum { [external\\_load\\_access](#) }
- typedef Size\_Type **size\_type**

### Public Member Functions

- [hash\\_load\\_check\\_resize\\_trigger](#) (float load\_min=0.125, float load\_max=0.5)
- [std::pair](#)< float, float > [get\\_loads](#) () const
- void [set\\_loads](#) ([std::pair](#)< float, float > load\_pair)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger](#) &other)

### Protected Member Functions

- bool **is\_grow\_needed** (size\_type size, size\_type num\_entries) const
- bool **is\_resize\_needed** () const
- void [notify\\_cleared](#) ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (size\_type num\_entries)
- void **notify\_externally\_resized** (size\_type new\_size)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void [notify\\_inserted](#) (size\_type num\_entries)
- void [notify\\_resized](#) (size\_type new\_size)

#### 5.357.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on a load check. It keeps the load factor between some load factors load\_min and load\_max.

Definition at line 175 of file hash\_policy.hpp.

#### 5.357.2 Member Enumeration Documentation

##### 5.357.2.1 anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

## Enumerator

<code>external_load_access</code>	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.
-----------------------------------	---

Definition at line 180 of file `hash_policy.hpp`.

## 5.357.3 Constructor & Destructor Documentation

### 5.357.3.1 `hash_load_check_resize_trigger()`

```
template<bool External_Load_Access, typename Size_Type >
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_resize_trigger
(
    float load_min = 0.125,
    float load_max = 0.5 )
```

Default constructor, or constructor taking `load_min` and `load_max` load factors between which this policy will keep the actual load.

Definition at line 47 of file `hash_policy.hpp`.

## 5.357.4 Member Function Documentation

### 5.357.4.1 `get_loads()`

```
template<bool External_Load_Access, typename Size_Type >
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::get_loads ( ) const [inline]
```

Returns a pair of the minimal and maximal loads, respectively.

Definition at line 236 of file `hash_policy.hpp`.

### 5.357.4.2 `notify_cleared()`

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ( ) [protected]
```

Notifies the table was cleared.

Definition at line 206 of file `hash_policy.hpp`.

#### 5.357.4.3 notify\_inserted()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
inserted (
    size_type num_entries ) [inline], [protected]
```

Notifies an element was inserted. The total number of entries in the table is num\_entries.

Definition at line 109 of file hash\_policy.hpp.

#### 5.357.4.4 notify\_resized()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
resized (
    size_type new_size ) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 151 of file hash\_policy.hpp.

#### 5.357.4.5 set\_loads()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
    std::pair< float, float > load_pair )
```

Sets the loads through a pair of the minimal and maximal loads, respectively.

Definition at line 245 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

### 5.358 \_\_gnu\_pbds::hash\_prime\_size\_policy Class Reference

#### Public Types

- typedef std::size\_t [size\\_type](#)

#### Public Member Functions

- [hash\\_prime\\_size\\_policy](#) ([size\\_type](#) start\_size=8)
- void **swap** ([hash\\_prime\\_size\\_policy](#) &other)

## Protected Member Functions

- `size_type` `get_nearest_larger_size` (`size_type` `size`) const
- `size_type` `get_nearest_smaller_size` (`size_type` `size`) const

### 5.358.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

Definition at line 450 of file `hash_policy.hpp`.

### 5.358.2 Member Typedef Documentation

#### 5.358.2.1 `size_type`

```
typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type
```

Size type.

Definition at line 454 of file `hash_policy.hpp`.

### 5.358.3 Constructor & Destructor Documentation

#### 5.358.3.1 `hash_prime_size_policy()`

```
__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (  
    size_type start_size = 8 ) [inline]
```

Default constructor, or onstructor taking a `start_size` The policy will use the sequence of sizes approximately `start_size`, `start_size* 2`, `start_size* 2^2`, ...

Definition at line 127 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- `hash_policy.hpp`

### 5.359 `__gnu_pbds::hash_standard_resize_policy`< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` > Class Template Reference

Inherits `Size_Policy`, and `Trigger_Policy`.

## Public Types

- enum { **external\_size\_access** }
- typedef Size\_Policy **size\_policy**
- typedef Size\_Type **size\_type**
- typedef Trigger\_Policy **trigger\_policy**

## Public Member Functions

- [hash\\_standard\\_resize\\_policy](#) ()
- [hash\\_standard\\_resize\\_policy](#) (const Size\_Policy &r\_size\_policy)
- [hash\\_standard\\_resize\\_policy](#) (const Size\_Policy &r\_size\_policy, const Trigger\_Policy &r\_trigger\_policy)
- size\_type [get\\_actual\\_size](#) () const
- Size\_Policy & [get\\_size\\_policy](#) ()
- const Size\_Policy & [get\\_size\\_policy](#) () const
- Trigger\_Policy & [get\\_trigger\\_policy](#) ()
- const Trigger\_Policy & [get\\_trigger\\_policy](#) () const
- void [resize](#) (size\_type suggested\_new\_size)
- void **swap** ([hash\\_standard\\_resize\\_policy](#)< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type > &other)

## Protected Member Functions

- size\_type [get\\_new\\_size](#) (size\_type size, size\_type num\_used\_e) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (size\_type num\_e)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** (size\_type num\_e)
- void **notify\_resized** (size\_type new\_size)

### 5.359.1 Detailed Description

```
template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_check_resize_↵
trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
```

A resize policy which delegates operations to size and trigger policies.

Definition at line 489 of file hash\_policy.hpp.

## 5.359.2 Constructor & Destructor Documentation

### 5.359.2.1 `hash_standard_resize_policy()` [1/3]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size←
_Type >::hash_standard_resize_policy ( )
```

Default constructor.

Definition at line 44 of file `hash_policy.hpp`.

### 5.359.2.2 `hash_standard_resize_policy()` [2/3]

```
template<typename Size_Policy, typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size←
_Type >::hash_standard_resize_policy (
    const Size_Policy & r_size_policy )
```

constructor taking some policies `r_size_policy` will be copied by the `Size_Policy` object of this object.

Definition at line 50 of file `hash_policy.hpp`.

### 5.359.2.3 `hash_standard_resize_policy()` [3/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size←
_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size←
_Type >::hash_standard_resize_policy (
    const Size_Policy & r_size_policy,
    const Trigger_Policy & r_trigger_policy )
```

constructor taking some policies. `r_size_policy` will be copied by the `Size_Policy` object of this object. `r_trigger_policy` will be copied by the `Trigger_Policy` object of this object.

Definition at line 56 of file `hash_policy.hpp`.

## 5.359.3 Member Function Documentation

#### 5.359.3.1 `get_actual_size()`

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↵
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↵
Access, Size_Type >::get_actual_size ( ) const [inline]
```

Returns the actual size of the container.

Definition at line 177 of file `hash_policy.hpp`.

#### 5.359.3.2 `get_new_size()`

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↵
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↵
Access, Size_Type >::get_new_size (
    size_type size,
    size_type num_used_e ) const [protected]
```

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

Definition at line 158 of file `hash_policy.hpp`.

#### 5.359.3.3 `get_size_policy()` [1/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_size_policy ( )
```

Access to the `Size_Policy` object used.

Definition at line 242 of file `hash_policy.hpp`.

#### 5.359.3.4 `get_size_policy()` [2/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
_Size_Access, Size_Type >::get_size_policy ( ) const
```

Const access to the `Size_Policy` object used.

Definition at line 248 of file `hash_policy.hpp`.

#### 5.359.3.5 get\_trigger\_policy() [1/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_trigger_policy ( )
```

Access to the Trigger\_Policy object used.

Definition at line 230 of file hash\_policy.hpp.

#### 5.359.3.6 get\_trigger\_policy() [2/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy,
External_Size_Access, Size_Type >::get_trigger_policy ( ) const
```

Access to the Trigger\_Policy object used.

Definition at line 236 of file hash\_policy.hpp.

#### 5.359.3.7 resize()

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
Size_Type >::resize (
    size_type suggested_new_size )
```

Resizes the container to suggested\_new\_size, a suggested size (the actual size will be determined by the Size\_Policy object).

Definition at line 186 of file hash\_policy.hpp.

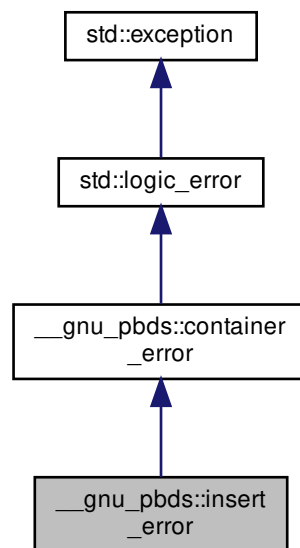
The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)



### 5.360 `__gnu_pbds::insert_error` Struct Reference

Inheritance diagram for `__gnu_pbds::insert_error`:



#### Public Member Functions

- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

#### 5.360.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

Definition at line 66 of file `exception.hpp`.

#### 5.360.2 Member Function Documentation

## 5.360.2.1 what()

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

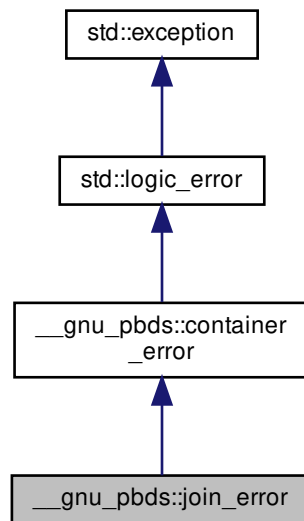
Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

## 5.361 \_\_gnu\_pbds::join\_error Struct Reference

Inheritance diagram for \_\_gnu\_pbds::join\_error:



## Public Member Functions

- virtual const char \* [what](#) ( ) const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

### 5.361.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

Definition at line 70 of file `exception.hpp`.

### 5.361.2 Member Function Documentation

#### 5.361.2.1 `what()`

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

## 5.362 `__gnu_pbds::linear_probe_fn< Size_Type >` Class Template Reference

### Public Types

- typedef `Size_Type` **size\_type**

### Public Member Functions

- void **swap** ([linear\\_probe\\_fn](#)< `Size_Type` > &other)

### Protected Member Functions

- `size_type` [operator\(\)](#) (`size_type` i) const

#### 5.362.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

Definition at line 61 of file `hash_policy.hpp`.

### 5.362.2 Member Function Documentation

#### 5.362.2.1 `operator()`

```
template<typename Size_Type >
linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (
    size_type i ) const [inline], [protected]
```

Returns the i-th offset from the hash value.

Definition at line 51 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

### 5.363 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

Inherits type< Key, Mapped, \_Alloc, list\_update\_tag, \_\_gnu\_cxx::typelist::create2< Eq\_Fn, Update\_Policy >::type >.

#### Public Types

- typedef [list\\_update\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Update\_Policy **update\_policy**

#### Public Member Functions

- template<typename It >  
[list\\_update](#) (It first, It last)
- **list\_update** (const [list\\_update](#) &other)
- [list\\_update](#) & **operator=** (const [list\\_update](#) &other)
- void **swap** ([list\\_update](#) &other)

#### 5.363.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy =
detail::default_update_policy::type, class _Alloc = std::allocator<char>>
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
```

A list-update based associative container.

**Template Parameters**

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Eq_Fn</i>	Equal functor.
<i>Update_Policy</i>	Update policy, determines when an element will be moved to the front of the list.
<i>_Alloc</i>	Allocator type.

Base is detail::lu\_map.

Definition at line 815 of file assoc\_container.hpp.

**5.363.2 Constructor & Destructor Documentation****5.363.2.1 list\_update()**

```
template<typename Key , typename Mapped , class Eq_Fn = typename detail::default_eq_fn<Key>↵
::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (
    It first,
    It last ) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

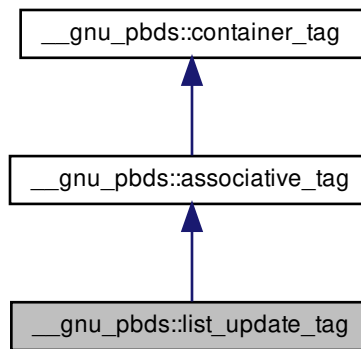
Definition at line 831 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.364 \_\_gnu\_pbds::list\_update\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::list\_update\_tag:



## 5.364.1 Detailed Description

List-update.

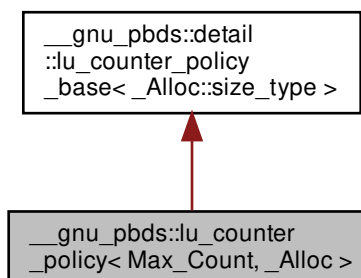
Definition at line 168 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.365 \_\_gnu\_pbds::lu\_counter\_policy&lt; Max\_Count, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >:



### Public Types

- enum { [max\\_count](#) }
- typedef `_Alloc` **allocator\_type**
- typedef `__rebind_m::other::reference` [metadata\\_reference](#)
- typedef `detail::lu_counter_metadata< size_type >` [metadata\\_type](#)
- typedef `allocator_type::size_type` **size\_type**

### Public Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) ([metadata\\_reference](#) r\_data) const

### Private Member Functions

- `lu_counter_metadata< size_type >` **operator()** (size\_type max\_size) const
- bool **operator()** (Metadata\_Reference r\_data, size\_type m\_max\_count) const

#### 5.365.1 Detailed Description

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

Definition at line 92 of file `list_update_policy.hpp`.

#### 5.365.2 Member Typedef Documentation

##### 5.365.2.1 [metadata\\_reference](#)

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>>
typedef __rebind_m::other::reference \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >::metadata\_reference
```

Reference to metadata on which this functor operates.

Definition at line 115 of file `list_update_policy.hpp`.

## 5.365.2.2 metadata\_type

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

Definition at line 107 of file list\_update\_policy.hpp.

## 5.365.3 Member Enumeration Documentation

## 5.365.3.1 anonymous enum

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
anonymous enum
```

## Enumerator

max_count	When some element is accessed this number of times, it will be moved to the front of the list.
-----------	--

Definition at line 99 of file list\_update\_policy.hpp.

## 5.365.4 Member Function Documentation

## 5.365.4.1 operator()() [1/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() ( ) const [inline]
```

Creates a metadata object.

Definition at line 119 of file list\_update\_policy.hpp.

References \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >::max\_count.



#### 5.365.4.2 `operator()` [2/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>>
bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() (
    metadata_reference r_data ) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

Definition at line 125 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

### 5.366 `__gnu_pbds::lu_move_to_front_policy< _Alloc >` Class Template Reference

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__rebind_m::other::reference` `metadata_reference`
- typedef `null_type` `metadata_type`

#### Public Member Functions

- `metadata_type operator() ()` const
- bool `operator() (metadata_reference r_metadata)` const

#### 5.366.1 Detailed Description

```
template<typename _Alloc = std::allocator<char>>>
class __gnu_pbds::lu_move_to_front_policy< _Alloc >
```

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

Definition at line 57 of file `list_update_policy.hpp`.

#### 5.366.2 Member Typedef Documentation

### 5.366.2.1 `metadata_reference`

```
template<typename _Alloc = std::allocator<char>>
typedef __rebind_m::other::reference __gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_reference
```

Reference to metadata on which this functor operates.

Definition at line 70 of file `list_update_policy.hpp`.

### 5.366.2.2 `metadata_type`

```
template<typename _Alloc = std::allocator<char>>
typedef null_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_type
```

Metadata on which this functor operates.

Definition at line 63 of file `list_update_policy.hpp`.

## 5.366.3 Member Function Documentation

### 5.366.3.1 `operator()()` [1/2]

```
template<typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator() ( ) const [inline]
```

Creates a metadata object.

Definition at line 74 of file `list_update_policy.hpp`.

### 5.366.3.2 `operator()()` [2/2]

```
template<typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator() (
    metadata_reference r_metadata ) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

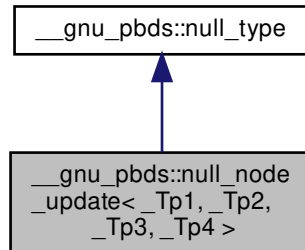
Definition at line 80 of file `list_update_policy.hpp`.

The documentation for this class was generated from the following file:

- `list_update_policy.hpp`

### 5.367 `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`:



#### 5.367.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>
struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
```

A null node updatator, indicating that no node updates are required.

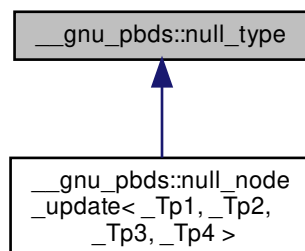
Definition at line 214 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.368 `__gnu_pbds::null_type` Struct Reference

Inheritance diagram for `__gnu_pbds::null_type`:



## 5.368.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

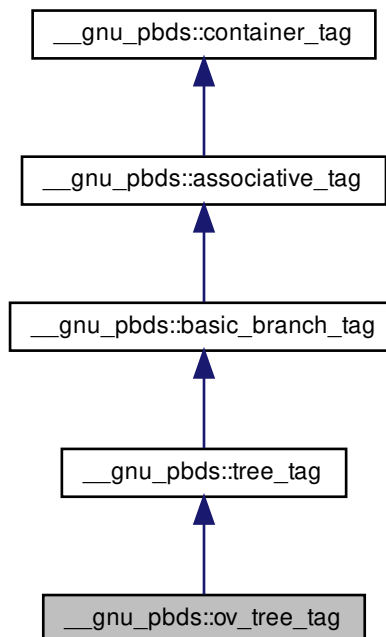
Definition at line 210 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.369 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::ov\_tree\_tag:



### 5.369.1 Detailed Description

Ordered-vector tree.

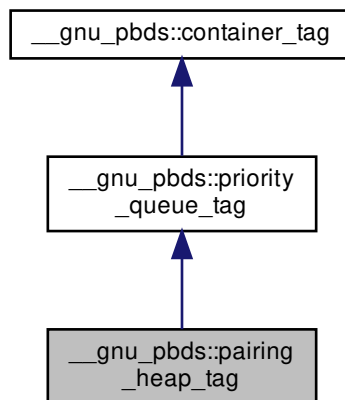
Definition at line 159 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.370 \_\_gnu\_pbds::pairing\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::pairing\_heap\_tag:



### 5.370.1 Detailed Description

Pairing-heap.

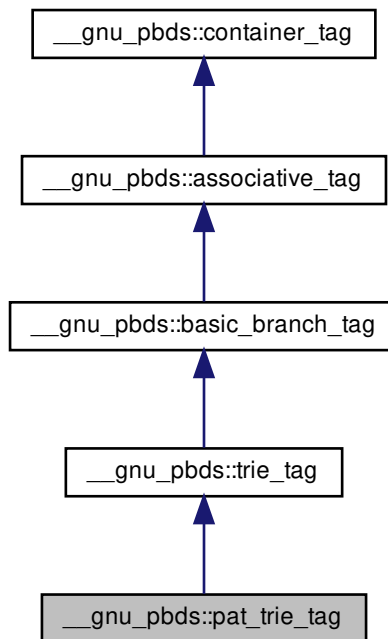
Definition at line 174 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.371 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



## 5.371.1 Detailed Description

PATRICIA trie.

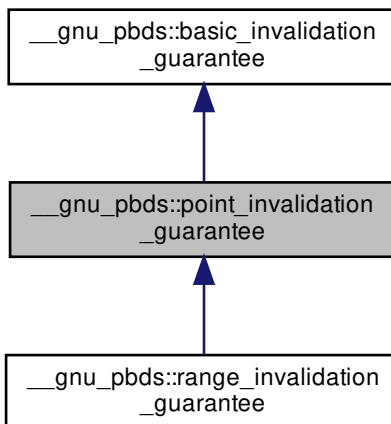
Definition at line 165 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.372 `__gnu_pbds::point_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::point_invalidation_guarantee`:



#### 5.372.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.373 `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>` Class Template Reference

Inherits `type<_Tv, Cmp_Fn, _Alloc, Tag>`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `Tag` **container\_category**
- typedef `allocator_type::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `allocator_type::size_type` **size\_type**
- typedef `_Tv` **value\_type**

## Public Member Functions

- [priority\\_queue](#) (const `cmp_fn` &`r_cmp_fn`)
- `template<typename It >`  
[priority\\_queue](#) (It `first_it`, It `last_it`)
- `template<typename It >`  
[priority\\_queue](#) (It `first_it`, It `last_it`, const `cmp_fn` &`r_cmp_fn`)
- **priority\_queue** (const [priority\\_queue](#) &`other`)
- [priority\\_queue](#) & **operator=** (const [priority\\_queue](#) &`other`)
- void **swap** ([priority\\_queue](#) &`other`)

## 5.373.1 Detailed Description

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>
class __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>
```

A priority queue composed of one specific heap policy.

## Template Parameters

<code>_Tv</code>	Value type.
<code>Cmp_Fn</code>	Comparison functor.
<code>Tag</code>	Instantiating data structure type, see <code>container_tag</code> .
<code>_Alloc</code>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.



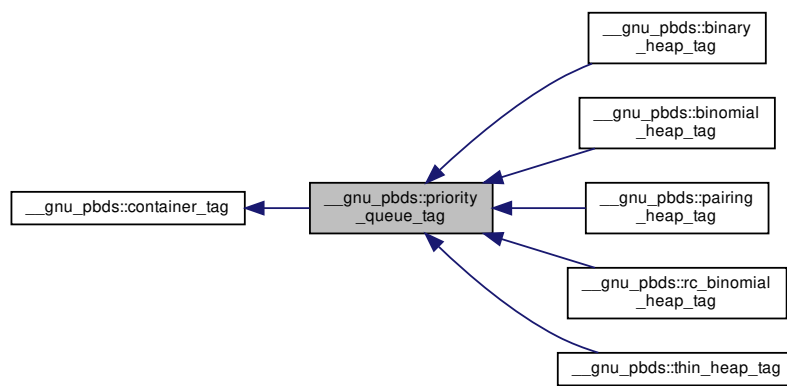
Definition at line 83 of file priority\_queue.hpp.

The documentation for this class was generated from the following file:

- [priority\\_queue.hpp](#)

### 5.374 \_\_gnu\_pbds::priority\_queue\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::priority\_queue\_tag:



#### 5.374.1 Detailed Description

Basic priority-queue.

Definition at line 171 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.375 \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type > Class Template Reference

#### Public Types

- typedef Size\_Type **size\_type**

#### Public Member Functions

- void **swap** ([quadratic\\_probe\\_fn](#)< Size\_Type > &other)

## Protected Member Functions

- `size_type operator() (size_type i) const`

## 5.375.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::quadratic_probe_fn< Size_Type >
```

A probe sequence policy using square increments.

Definition at line 85 of file `hash_policy.hpp`.

## 5.375.2 Member Function Documentation

5.375.2.1 `operator()()`

```
template<typename Size_Type >
quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator()
(
    size_type i ) const [inline], [protected]
```

Returns the i-th offset from the hash value.

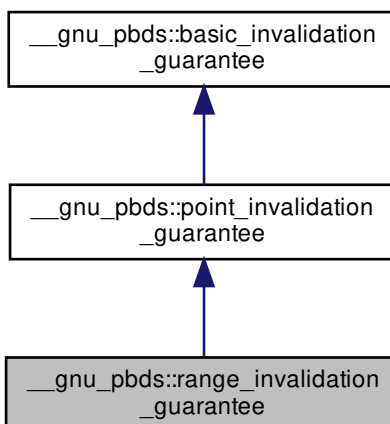
Definition at line 51 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

5.376 `__gnu_pbds::range_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::range_invalidation_guarantee`:



### 5.376.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

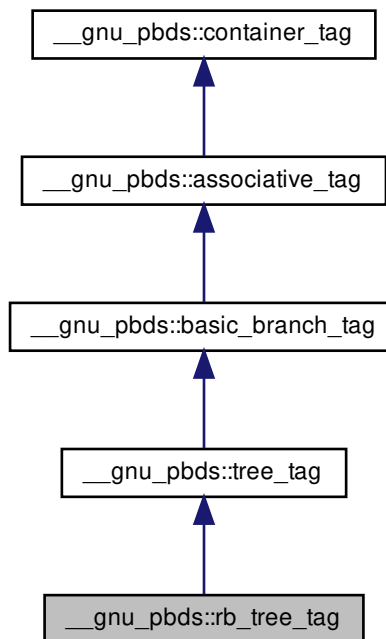
Definition at line 114 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.377 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

Inheritance diagram for `__gnu_pbds::rb_tree_tag`:



### 5.377.1 Detailed Description

Red-black tree.

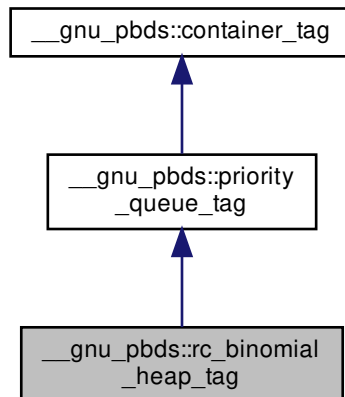
Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

5.378 `__gnu_pbds::rc_binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::rc_binomial_heap_tag`:



## 5.378.1 Detailed Description

Redundant-counter binomial-heap.

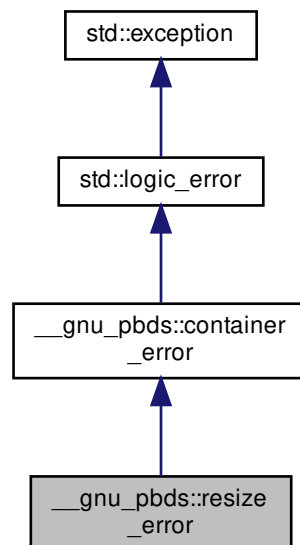
Definition at line 180 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.379 `__gnu_pbds::resize_error` Struct Reference

Inheritance diagram for `__gnu_pbds::resize_error`:



#### Public Member Functions

- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

#### 5.379.1 Detailed Description

A container cannot be resized.

Definition at line 73 of file `exception.hpp`.

#### 5.379.2 Member Function Documentation

## 5.379.2.1 what()

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

## 5.380 \_\_gnu\_pbds::sample\_probe\_fn Class Reference

## Public Types

- typedef std::size\_t **size\_type**

## Public Member Functions

- [sample\\_probe\\_fn](#) ()
- [sample\\_probe\\_fn](#) (const [sample\\_probe\\_fn](#) &)
- void [swap](#) ([sample\\_probe\\_fn](#) &)

## Protected Member Functions

- size\_type [operator\(\)](#) (key\_const\_reference r\_key, size\_type i) const

## 5.380.1 Detailed Description

A sample probe policy.

Definition at line 47 of file [sample\\_probe\\_fn.hpp](#).

## 5.380.2 Constructor &amp; Destructor Documentation

5.380.2.1 [sample\\_probe\\_fn](#)() [1/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn ( )
```

Default constructor.

### 5.380.2.2 `sample_probe_fn()` [2/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn (  
    const sample\_probe\_fn & )
```

Copy constructor.

## 5.380.3 Member Function Documentation

### 5.380.3.1 `operator()()`

```
size_type __gnu_pbds::sample_probe_fn::operator() (  
    key_const_reference r_key,  
    size_type i ) const [inline], [protected]
```

Returns the *i*-th offset from the hash value of some key *r\_key*.

### 5.380.3.2 `swap()`

```
void __gnu_pbds::sample_probe_fn::swap (  
    sample\_probe\_fn & ) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_probe\\_fn.hpp](#)

## 5.381 `__gnu_pbds::sample_range_hashing` Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_range\\_hashing](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_range\\_hashing](#) &other)
- void [swap](#) ([sample\\_range\\_hashing](#) &other)

### Protected Member Functions

- void [notify\\_resized](#) ([size\\_type](#))
- [size\\_type](#) [operator\(\)](#) ([size\\_type](#)) const

#### 5.381.1 Detailed Description

A sample range-hashing functor.

Definition at line 47 of file `sample_range_hashing.hpp`.

#### 5.381.2 Member Typedef Documentation

##### 5.381.2.1 `size_type`

```
typedef std::size_t __gnu_pbds::sample_range_hashing::size_type
```

Size type.

Definition at line 51 of file `sample_range_hashing.hpp`.

#### 5.381.3 Constructor & Destructor Documentation

##### 5.381.3.1 `sample_range_hashing()` [1/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing ( )
```

Default constructor.

##### 5.381.3.2 `sample_range_hashing()` [2/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing (
    const sample\_range\_hashing & other )
```

Copy constructor.

#### 5.381.4 Member Function Documentation



#### 5.381.4.1 `notify_resized()`

```
void __gnu_pbds::sample_range_hashing::notify_resized (
    size_type ) [protected]
```

Notifies the policy object that the container's size has changed to argument's size.

#### 5.381.4.2 `operator()()`

```
size_type __gnu_pbds::sample_range_hashing::operator() (
    size_type ) const [inline], [protected]
```

Transforms the `__hash` value hash into a ranged-hash value.

#### 5.381.4.3 `swap()`

```
void __gnu_pbds::sample_range_hashing::swap (
    sample_range_hashing & other ) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_range\\_hashing.hpp](#)

### 5.382 `__gnu_pbds::sample_ranged_hash_fn` Class Reference

#### Public Types

- typedef std::size\_t **size\_type**

#### Public Member Functions

- [sample\\_ranged\\_hash\\_fn](#) ()
- [sample\\_ranged\\_hash\\_fn](#) (const [sample\\_ranged\\_hash\\_fn](#) &)
- void [swap](#) ([sample\\_ranged\\_hash\\_fn](#) &)

#### Protected Member Functions

- void [notify\\_resized](#) (size\_type)
- size\_type [operator\(\)](#) (key\_const\_reference) const

### 5.382.1 Detailed Description

A sample ranged-hash functor.

Definition at line 47 of file sample\_ranged\_hash\_fn.hpp.

### 5.382.2 Constructor & Destructor Documentation

#### 5.382.2.1 sample\_ranged\_hash\_fn() [1/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ( )
```

Default constructor.

#### 5.382.2.2 sample\_ranged\_hash\_fn() [2/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (
    const sample_ranged_hash_fn & )
```

Copy constructor.

### 5.382.3 Member Function Documentation

#### 5.382.3.1 notify\_resized()

```
void __gnu_pbds::sample_ranged_hash_fn::notify_resized (
    size_type ) [protected]
```

Notifies the policy object that the container's \_\_size has changed to size.

#### 5.382.3.2 operator>()

```
size_type __gnu_pbds::sample_ranged_hash_fn::operator() (
    key_const_reference ) const [inline], [protected]
```

Transforms key\_const\_reference into a position within the table.

### 5.382.3.3 swap()

```
void __gnu_pbds::sample_ranged_hash_fn::swap (
    sample_ranged_hash_fn & ) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_hash\\_fn.hpp](#)

## 5.383 \_\_gnu\_pbds::sample\_ranged\_probe\_fn Class Reference

### Public Types

- typedef std::size\_t **size\_type**

### Public Member Functions

- **sample\_ranged\_probe\_fn** (const [sample\\_ranged\\_probe\\_fn](#) &)
- void **swap** ([sample\\_ranged\\_probe\\_fn](#) &)

### Protected Member Functions

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference, std::size\_t, size\_type) const

### 5.383.1 Detailed Description

A sample ranged-probe functor.

Definition at line 47 of file [sample\\_ranged\\_probe\\_fn.hpp](#).

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_probe\\_fn.hpp](#)

## 5.384 \_\_gnu\_pbds::sample\_resize\_policy Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_policy](#) &other)
- void [swap](#) ([sample\\_resize\\_policy](#) &other)

### Protected Member Functions

- [size\\_type](#) [get\\_new\\_size](#) ([size\\_type](#) size, [size\\_type](#) num\_used\_e) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_e)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_e)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

#### 5.384.1 Detailed Description

A sample resize policy.

Definition at line 47 of file `sample_resize_policy.hpp`.

#### 5.384.2 Member Typedef Documentation

##### 5.384.2.1 `size_type`

```
typedef std::size_t __gnu_pbds::sample_resize_policy::size_type
```

Size type.

Definition at line 51 of file `sample_resize_policy.hpp`.

#### 5.384.3 Constructor & Destructor Documentation

#### 5.384.3.1 sample\_resize\_policy()

```
__gnu_pbds::sample_resize_policy::sample_resize_policy ( )
```

Default constructor.

### 5.384.4 Member Function Documentation

#### 5.384.4.1 get\_new\_size()

```
size_type __gnu_pbds::sample_resize_policy::get_new_size (
    size_type size,
    size_type num_used_e ) const    [protected]
```

Queries what the new size should be.

#### 5.384.4.2 is\_resize\_needed()

```
bool __gnu_pbds::sample_resize_policy::is_resize_needed ( ) const    [inline], [protected]
```

Queries whether a resize is needed.

#### 5.384.4.3 notify\_cleared()

```
void __gnu_pbds::sample_resize_policy::notify_cleared ( )    [protected]
```

Notifies the table was cleared.

#### 5.384.4.4 notify\_erase\_search\_collision()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_collision ( )    [inline], [protected]
```

Notifies a search encountered a collision.

#### 5.384.4.5 notify\_erase\_search\_end()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_end ( )    [inline], [protected]
```

Notifies a search ended.

#### 5.384.4.6 notify\_erase\_search\_start()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_start ( ) [inline], [protected]
```

Notifies a search started.

#### 5.384.4.7 notify\_erased()

```
void __gnu_pbds::sample_resize_policy::notify_erased (
    size_type num_e ) [inline], [protected]
```

Notifies an element was erased.

#### 5.384.4.8 notify\_find\_search\_collision()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

#### 5.384.4.9 notify\_find\_search\_end()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_end ( ) [inline], [protected]
```

Notifies a search ended.

#### 5.384.4.10 notify\_find\_search\_start()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_start ( ) [inline], [protected]
```

Notifies a search started.

#### 5.384.4.11 notify\_insert\_search\_collision()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

**5.384.4.12 notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_end ( ) [inline], [protected]
```

Notifies a search ended.

**5.384.4.13 notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_start ( ) [inline], [protected]
```

Notifies a search started.

**5.384.4.14 notify\_inserted()**

```
void __gnu_pbds::sample_resize_policy::notify_inserted (
    size_type num_e ) [inline], [protected]
```

Notifies an element was inserted.

**5.384.4.15 notify\_resized()**

```
void __gnu_pbds::sample_resize_policy::notify_resized (
    size_type new_size ) [protected]
```

Notifies the table was resized to new\_size.

**5.384.4.16 sample\_range\_hashing()**

```
__gnu_pbds::sample_resize_policy::sample_range_hashing (
    const sample_resize_policy & other )
```

Copy constructor.

**5.384.4.17 swap()**

```
void __gnu_pbds::sample_resize_policy::swap (
    sample_resize_policy & other ) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_policy.hpp](#)

## 5.385 `__gnu_pbds::sample_resize_trigger` Class Reference

### Public Types

- typedef `std::size_t` `size_type`

### Public Member Functions

- `sample_resize_trigger` ()
- `sample_range_hashing` (const `sample_resize_trigger` &)
- void `swap` (`sample_resize_trigger` &)

### Protected Member Functions

- bool `is_grow_needed` (`size_type` size, `size_type` num\_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (`size_type` num\_entries)
- void `notify_externally_resized` (`size_type` new\_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (`size_type` num\_entries)
- void `notify_resized` (`size_type` new\_size)

#### 5.385.1 Detailed Description

A sample resize trigger policy.

Definition at line 47 of file `sample_resize_trigger.hpp`.

#### 5.385.2 Member Typedef Documentation

##### 5.385.2.1 `size_type`

```
typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type
```

Size type.

Definition at line 51 of file `sample_resize_trigger.hpp`.



### 5.385.3 Constructor & Destructor Documentation

#### 5.385.3.1 sample\_resize\_trigger()

```
__gnu_pbds::sample_resize_trigger::sample_resize_trigger ( )
```

Default constructor.

### 5.385.4 Member Function Documentation

#### 5.385.4.1 is\_grow\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_grow_needed (
    size_type size,
    size_type num_entries ) const [inline], [protected]
```

Queries whether a grow is needed.

#### 5.385.4.2 is\_resize\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_resize_needed ( ) const [inline], [protected]
```

Queries whether a resize is needed.

#### 5.385.4.3 notify\_cleared()

```
void __gnu_pbds::sample_resize_trigger::notify_cleared ( ) [protected]
```

Notifies the table was cleared.

#### 5.385.4.4 notify\_erase\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

#### 5.385.4.5 notify\_erase\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_end ( ) [inline], [protected]
```

Notifies a search ended.

#### 5.385.4.6 notify\_erase\_search\_start()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_start ( ) [inline], [protected]
```

Notifies a search started.

#### 5.385.4.7 notify\_erased()

```
void __gnu_pbds::sample_resize_trigger::notify_erased (
    size_type num_entries ) [inline], [protected]
```

Notifies an element was erased.

#### 5.385.4.8 notify\_externally\_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_externally_resized (
    size_type new_size ) [protected]
```

Notifies the table was resized externally.

#### 5.385.4.9 notify\_find\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

#### 5.385.4.10 notify\_find\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_end ( ) [inline], [protected]
```

Notifies a search ended.

**5.385.4.11 notify\_find\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_start ( ) [inline], [protected]
```

Notifies a search started.

**5.385.4.12 notify\_insert\_search\_collision()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision ( ) [inline], [protected]
```

Notifies a search encountered a collision.

**5.385.4.13 notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_end ( ) [inline], [protected]
```

Notifies a search ended.

**5.385.4.14 notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_start ( ) [inline], [protected]
```

Notifies a search started.

**5.385.4.15 notify\_inserted()**

```
void __gnu_pbds::sample_resize_trigger::notify_inserted (
    size_type num_entries ) [inline], [protected]
```

Notifies an element was inserted. the total number of entries in the table is num\_entries.

**5.385.4.16 notify\_resized()**

```
void __gnu_pbds::sample_resize_trigger::notify_resized (
    size_type new_size ) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

#### 5.385.4.17 sample\_range\_hashing()

```
__gnu_pbds::sample_resize_trigger::sample_range_hashing (  
    const sample\_resize\_trigger & )
```

Copy constructor.

#### 5.385.4.18 swap()

```
void __gnu_pbds::sample_resize_trigger::swap (  
    sample\_resize\_trigger & ) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_trigger.hpp](#)

## 5.386 \_\_gnu\_pbds::sample\_size\_policy Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_size\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_size\\_policy](#) &)
- void [swap](#) ([sample\\_size\\_policy](#) &other)

### Protected Member Functions

- [size\\_type](#) [get\\_nearest\\_larger\\_size](#) ([size\\_type](#) size) const
- [size\\_type](#) [get\\_nearest\\_smaller\\_size](#) ([size\\_type](#) size) const

#### 5.386.1 Detailed Description

A sample size policy.

Definition at line 47 of file [sample\\_size\\_policy.hpp](#).

#### 5.386.2 Member Typedef Documentation

#### 5.386.2.1 size\_type

```
typedef std::size_t __gnu_pbds::sample_size_policy::size_type
```

Size type.

Definition at line 51 of file sample\_size\_policy.hpp.

### 5.386.3 Constructor & Destructor Documentation

#### 5.386.3.1 sample\_size\_policy()

```
__gnu_pbds::sample_size_policy::sample_size_policy ( )
```

Default constructor.

### 5.386.4 Member Function Documentation

#### 5.386.4.1 get\_nearest\_larger\_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size (
    size_type size ) const [inline], [protected]
```

Given a \_\_size size, returns a \_\_size that is larger.

#### 5.386.4.2 get\_nearest\_smaller\_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size (
    size_type size ) const [inline], [protected]
```

Given a \_\_size size, returns a \_\_size that is smaller.

#### 5.386.4.3 sample\_range\_hashing()

```
__gnu_pbds::sample_size_policy::sample_range_hashing (
    const sample_size_policy & )
```

Copy constructor.

#### 5.386.4.4 `swap()`

```
void __gnu_pbds::sample_size_policy::swap (
    sample_size_policy & other ) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_size\\_policy.hpp](#)

### 5.387 `__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >` Class Template Reference

#### 5.387.1 Detailed Description

```
template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >
```

A sample node updatator.

Definition at line 49 of file `sample_tree_node_update.hpp`.

The documentation for this class was generated from the following file:

- [sample\\_tree\\_node\\_update.hpp](#)

### 5.388 `__gnu_pbds::sample_trie_access_traits` Struct Reference

#### Public Types

- enum { **max\_size** }
- typedef `_Alloc::template rebind< key_type >` **\_\_rebind\_k**
- typedef `std::string::const_iterator` **const\_iterator**
- typedef char **e\_type**
- typedef `__rebind_k::other::const_reference` **key\_const\_reference**
- typedef `std::string` **key\_type**
- typedef `std::size_t` **size\_type**

#### Static Public Member Functions

- static `const_iterator` **begin** (`key_const_reference`)
- static `size_type` **e\_pos** (`e_type`)
- static `const_iterator` **end** (`key_const_reference`)

### 5.388.1 Detailed Description

A sample trie element access traits.

Definition at line 47 of file `sample_trie_access_traits.hpp`.

### 5.388.2 Member Typedef Documentation

#### 5.388.2.1 `e_type`

```
typedef char __gnu_pbds::sample_trie_access_traits::e_type
```

Element type.

Definition at line 57 of file `sample_trie_access_traits.hpp`.

### 5.388.3 Member Function Documentation

#### 5.388.3.1 `begin()`

```
static const_iterator __gnu_pbds::sample_trie_access_traits::begin (  
    key_const_reference ) [inline], [static]
```

Returns a `const_iterator` to the first element of `r_key`.

#### 5.388.3.2 `e_pos()`

```
static size_type __gnu_pbds::sample_trie_access_traits::e_pos (  
    e_type ) [inline], [static]
```

Maps an element to a position.

#### 5.388.3.3 `end()`

```
static const_iterator __gnu_pbds::sample_trie_access_traits::end (  
    key_const_reference ) [inline], [static]
```

Returns a `const_iterator` to the after-last element of `r_key`.

The documentation for this struct was generated from the following file:

- [sample\\_trie\\_access\\_traits.hpp](#)

## 5.389 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

### Public Types

- typedef `std::size_t` **metadata\_type**

### Protected Member Functions

- [sample\\_trie\\_node\\_update](#) ()
- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

### 5.389.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

Definition at line 49 of file `sample_trie_node_update.hpp`.

### 5.389.2 Constructor & Destructor Documentation

#### 5.389.2.1 `sample_trie_node_update()`

```
template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_update
( ) [protected]
```

Default constructor.

### 5.389.3 Member Function Documentation

#### 5.389.3.1 `operator()`

```
template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
void __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
    node_iterator ,
    node_const_iterator ) const [inline], [protected]
```

Updates the rank of a node through a node\_iterator `node_it`; `end_nd_it` is the end node iterator.

The documentation for this class was generated from the following file:

- [sample\\_trie\\_node\\_update.hpp](#)



## 5.390 `__gnu_pbds::sample_update_policy` Struct Reference

### Public Member Functions

- [sample\\_update\\_policy](#) ()
- [sample\\_update\\_policy](#) (const [sample\\_update\\_policy](#) &)
- void [swap](#) ([sample\\_update\\_policy](#) &other)

### Protected Types

- typedef some\_metadata\_type [metadata\\_type](#)

### Protected Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) (metadata\_reference) const

#### 5.390.1 Detailed Description

A sample list-update policy.

Definition at line 47 of file `sample_update_policy.hpp`.

#### 5.390.2 Member Typedef Documentation

##### 5.390.2.1 `metadata_type`

```
typedef some_metadata_type \_\_gnu\_pbds::sample\_update\_policy::metadata\_type [protected]
```

Metadata on which this functor operates.

Definition at line 61 of file `sample_update_policy.hpp`.

#### 5.390.3 Constructor & Destructor Documentation

##### 5.390.3.1 `sample_update_policy()` [1/2]

```
\_\_gnu\_pbds::sample\_update\_policy::sample\_update\_policy ( )
```

Default constructor.

### 5.390.3.2 sample\_update\_policy() [2/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy (  
    const sample\_update\_policy & )
```

Copy constructor.

## 5.390.4 Member Function Documentation

### 5.390.4.1 operator()() [1/2]

```
metadata\_type __gnu_pbds::sample_update_policy::operator() ( ) const [protected]
```

Creates a metadata object.

### 5.390.4.2 operator()() [2/2]

```
bool __gnu_pbds::sample_update_policy::operator() (  
    metadata\_reference ) const [protected]
```

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

### 5.390.4.3 swap()

```
void __gnu_pbds::sample_update_policy::swap (  
    sample\_update\_policy & other ) [inline]
```

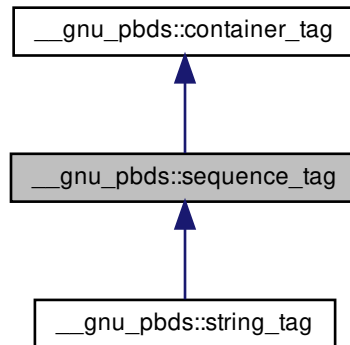
Swaps content.

The documentation for this struct was generated from the following file:

- [sample\\_update\\_policy.hpp](#)

### 5.391 `__gnu_pbds::sequence_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::sequence_tag`:



#### 5.391.1 Detailed Description

Basic sequence.

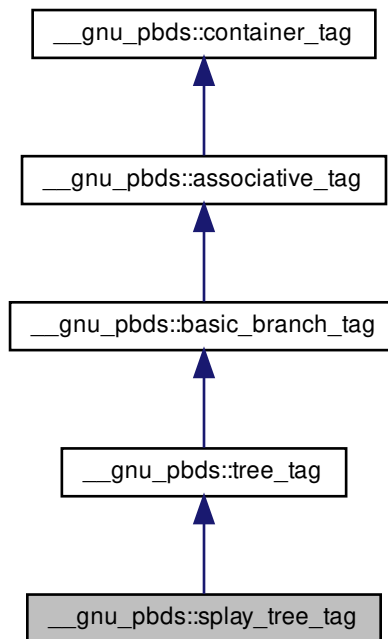
Definition at line 129 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.392 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::splay\_tree\_tag:



## 5.392.1 Detailed Description

Splay tree.

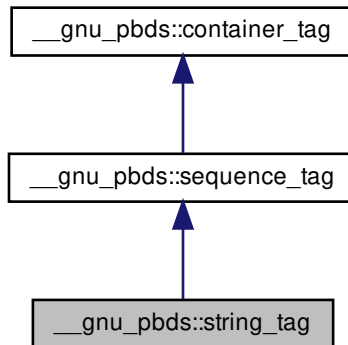
Definition at line 156 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.393 `__gnu_pbds::string_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::string_tag`:



#### 5.393.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

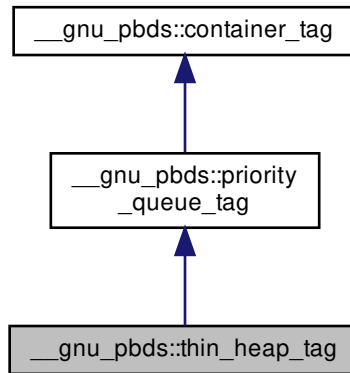
Definition at line 132 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.394 \_\_gnu\_pbds::thin\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::thin\_heap\_tag:



## 5.394.1 Detailed Description

Thin heap.

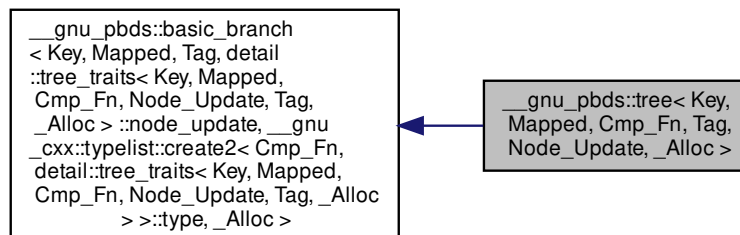
Definition at line 186 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.395 \_\_gnu\_pbds::tree&lt; Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >:



## Public Types

- typedef Cmp\_Fn [cmp\\_fn](#)
- typedef [detail::tree\\_traits](#)< Key, Mapped, Cmp\_Fn, Node\_Update, Tag, \_Alloc > ::node\_update **node\_update**

## Public Member Functions

- [tree](#) (const [cmp\\_fn](#) &c)
- template<typename It >  
[tree](#) (It first, It last)
- template<typename It >  
[tree](#) (It first, It last, const [cmp\\_fn](#) &c)
- **tree** (const [tree](#) &other)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

## 5.395.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename
Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update = null_node_update, typename _Alloc =
std::allocator<char>>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
```

A tree-based container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Cmp_Fn</i>	Comparison functor.
<i>Tag</i>	Instantiating data structure type, see <a href="#">container_tag</a> .
<i>Node_Update</i>	Updates tree internal-nodes, restores invariants when invalidated. XXX See <a href="#">design::tree-based-containersnode</a> invariants.
<i>_Alloc</i>	Allocator type.

Base tag choices are: [ov\\_tree\\_tag](#), [rb\\_tree\\_tag](#), [splay\\_tree\\_tag](#).

Base is [basic\\_branch](#).

Definition at line 635 of file [assoc\\_container.hpp](#).

## 5.395.2 Member Typedef Documentation

### 5.395.2.1 cmp\_fn

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn
```

Comparison functor type.

Definition at line 642 of file assoc\_container.hpp.

### 5.395.3 Constructor & Destructor Documentation

#### 5.395.3.1 tree() [1/3]

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
    const cmp_fn & c ) [inline]
```

Constructor taking some policy objects. r\_cmp\_fn will be copied by the Cmp\_Fn object of the container object.

Definition at line 648 of file assoc\_container.hpp.

#### 5.395.3.2 tree() [2/3]

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
    It first,
    It last ) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

Definition at line 655 of file assoc\_container.hpp.



5.395.3.3 `tree()` [3/3]

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb<
_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>>
template<typename It >
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
    It first,
    It last,
    const cmp_fn & c ) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_cmp\_fn will be copied by the cmp\_fn object of the container object.

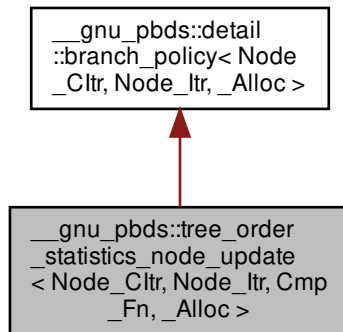
Definition at line 663 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

5.396 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

#### Public Member Functions

- const\_iterator `find_by_order` (`size_type`) const
- iterator `find_by_order` (`size_type`)
- `size_type` `order_of_key` (`key_const_reference`) const

#### Protected Member Functions

- void `operator()` (`node_iterator`, `node_const_iterator`) const

#### Private Types

- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rckey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

#### Private Member Functions

- virtual `it_type` **end** ()=0
- `it_type` **end\_iterator** () const

#### Static Private Member Functions

- static `key_const_reference` **extract\_key** (`const_reference` `r_val`)

#### 5.396.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >
```

Functor updating ranks of entrees.

Definition at line 64 of file `tree_policy.hpp`.

#### 5.396.2 Member Function Documentation

#### 5.396.2.1 `find_by_order()` [1/2]

```
template<typename Node_CIter , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
tree_order_statistics_node_update< Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::const_iterator __gnu_pbds::tree_order_
Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
    size_type order ) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 72 of file `tree_policy.hpp`.

#### 5.396.2.2 `find_by_order()` [2/2]

```
template<typename Node_CIter , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
tree_order_statistics_node_update< Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::iterator __gnu_pbds::tree_order_statist
Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
    size_type order ) [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file `tree_policy.hpp`.

#### 5.396.2.3 `operator()()`

```
template<typename Node_CIter , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
void __gnu_pbds::tree_order_statistics_node_update< Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::operator()
(
    node_iterator node_it,
    node_const_iterator end_nd_it ) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 108 of file `tree_policy.hpp`.

#### 5.396.2.4 `order_of_key()`

```
template<typename Node_CIter , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
tree_order_statistics_node_update< Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::size_type __gnu_pbds::tree_order_statist
Node_CIter, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (
    key_const_reference r_key ) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

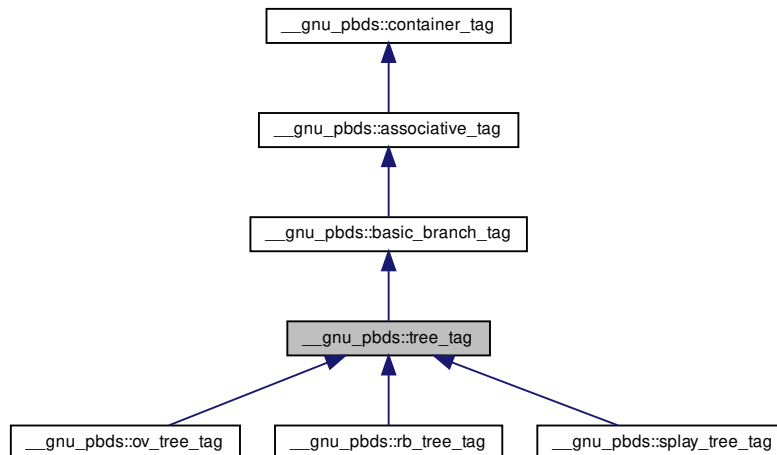
Definition at line 78 of file `tree_policy.hpp`.

The documentation for this class was generated from the following file:

- [tree\\_policy.hpp](#)

## 5.397 \_\_gnu\_pbds::tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::tree\_tag:



## 5.397.1 Detailed Description

Basic tree structure.

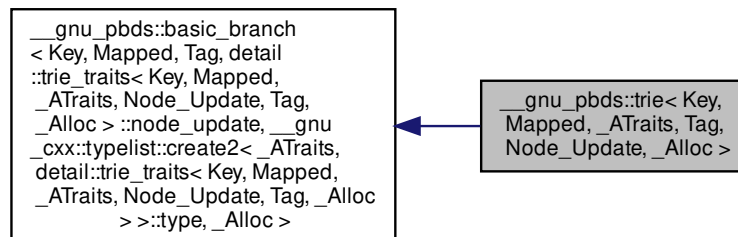
Definition at line 150 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.398 \_\_gnu\_pbds::trie&lt; Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >:



## Public Types

- typedef `_ATraits` [access\\_traits](#)
- typedef [detail::trie\\_traits](#)< Key, Mapped, `_ATraits`, Node\_Update, Tag, `_Alloc` > ::node\_update **node\_update**

## Public Member Functions

- [trie](#) (const [access\\_traits](#) &t)
- template<typename It >  
  [trie](#) (It first, It last)
- template<typename It >  
  [trie](#) (It first, It last, const [access\\_traits](#) &t)
- **trie** (const [trie](#) &other)
- [trie](#) & **operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

## 5.398.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename
Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update
= null_node_update, typename _Alloc = std::allocator<char>>>
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >
```

A trie-based container.

## Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>_ATraits</i>	Element access traits.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Node_Update</i>	Updates trie internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containersnode</code> invariants.
<i>_Alloc</i>	Allocator type.

Base tag choice is `pat_trie_tag`.

Base is `basic_branch`.

Definition at line 731 of file `assoc_container.hpp`.

## 5.398.2 Member Typedef Documentation

### 5.398.2.1 access\_traits

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵  
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename  
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename  
_Alloc = std::allocator<char>>>  
typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits
```

Element access traits type.

Definition at line 738 of file assoc\_container.hpp.

### 5.398.3 Constructor & Destructor Documentation

#### 5.398.3.1 trie() [1/3]

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵  
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename  
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename  
_Alloc = std::allocator<char>>>  
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (  
    const access_traits & t ) [inline]
```

Constructor taking some policy objects. r\_access\_traits will be copied by the \_ATraits object of the container object.

Definition at line 744 of file assoc\_container.hpp.

#### 5.398.3.2 trie() [2/3]

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵  
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename  
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename  
_Alloc = std::allocator<char>>>  
template<typename It >  
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (  
    It first,  
    It last ) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

Definition at line 751 of file assoc\_container.hpp.

5.398.3.3 `trie()` [3/3]

```

template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename
_Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
    It first,
    It last,
    const access_traits & t ) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

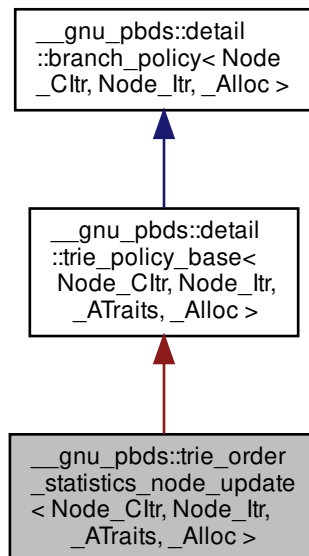
Definition at line 758 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.399 `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



## Public Types

- typedef `access_traits::const_iterator` **a\_const\_iterator**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Public Member Functions

- `const_iterator` [find\\_by\\_order](#) (`size_type`) const
- `iterator` [find\\_by\\_order](#) (`size_type`)
- `size_type` [order\\_of\\_key](#) (`key_const_reference`) const
- `size_type` [order\\_of\\_prefix](#) (`a_const_iterator`, `a_const_iterator`) const

## Protected Member Functions

- void [operator\(\)](#) (`node_iterator`, `node_const_iterator`) const

## Private Types

- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Private Member Functions

- virtual `const_iterator` **end** () const =0
- `it_type` **end\_iterator** () const
- virtual const `access_traits` & **get\_access\_traits** () const =0

## Static Private Member Functions

- static `size_type` **common\_prefix\_len** (`node_iterator`, `e_const_iterator`, `e_const_iterator`, const `access_traits` &)
- static `key_const_reference` **extract\_key** (const `reference` r\_val)
- static `iterator` **leftmost\_it** (`node_iterator`)
- static bool **less** (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, const `access_traits` &)
- static `iterator` **rightmost\_it** (`node_iterator`)



### 5.399.1 Detailed Description

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >
```

Functor updating ranks of entrees.

Definition at line 253 of file trie\_policy.hpp.

### 5.399.2 Member Function Documentation

#### 5.399.2.1 find\_by\_order() [1/2]

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::const_iterator __gnu_pbds::trie_order_statistics_node_update<
Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
    size_type order ) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 79 of file trie\_policy.hpp.

#### 5.399.2.2 find\_by\_order() [2/2]

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::iterator __gnu_pbds::trie_order_statistics_node_update<
Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
    size_type order ) [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file trie\_policy.hpp.

#### 5.399.2.3 operator()()

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
void __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::operator() (
    node_iterator nd_it,
    node_const_iterator ) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 152 of file trie\_policy.hpp.

#### 5.399.2.4 order\_of\_key()

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::order_of_key (
    key_const_reference r_key ) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

Definition at line 85 of file `trie_policy.hpp`.

#### 5.399.2.5 order\_of\_prefix()

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
    a_const_iterator b,
    a_const_iterator e ) const [inline]
```

Returns the order of a prefix within a sequence. For example, if `[b, e]` is the smallest prefix, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

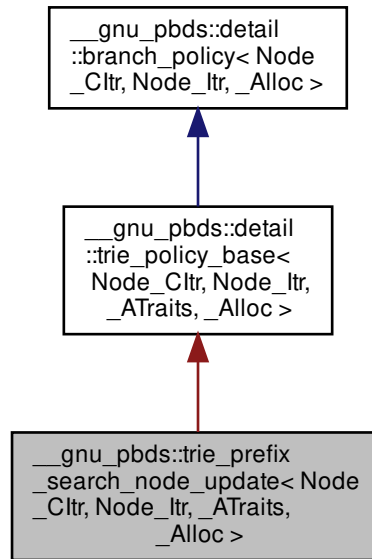
Definition at line 96 of file `trie_policy.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

## 5.400 `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



### Public Types

- typedef `access_traits::const_iterator` [a\\_const\\_iterator](#)
- typedef `_ATraits` [access\\_traits](#)
- typedef `_Alloc` [allocator\\_type](#)
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` [metadata\\_type](#)
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` [size\\_type](#)

### Public Member Functions

- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) (`key_const_reference`) const
- `std::pair< iterator, iterator >` [prefix\\_range](#) (`key_const_reference`)
- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#)) const
- `std::pair< iterator, iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#))

### Protected Member Functions

- void `operator()` (node\_iterator node\_it, node\_const\_iterator end\_nd\_it) const

### Private Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind\_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

### Private Member Functions

- `it_type` **end\_iterator** () const

### Static Private Member Functions

- static `size_type` **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const `access_traits` &)
- static `key_const_reference` **extract\_key** (const\_reference r\_val)
- static `iterator` **leftmost\_it** (node\_iterator)
- static `bool` **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const `access_traits` &)
- static `iterator` **rightmost\_it** (node\_iterator)

#### 5.400.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A node updator that allows tries to be searched for the range of values that match a certain prefix.

Definition at line 155 of file `trie_policy.hpp`.

#### 5.400.2 Member Typedef Documentation

#### 5.400.2.1 `a_const_iterator`

```
template<typename Node_CIter, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_CIter,
Node_Itr, _ATraits, _Alloc >::a_const_iterator
```

Const element iterator.

Definition at line 168 of file `trie_policy.hpp`.

#### 5.400.2.2 `access_traits`

```
template<typename Node_CIter, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _↵
Alloc >::access_traits
```

Element access traits.

Definition at line 165 of file `trie_policy.hpp`.

#### 5.400.2.3 `allocator_type`

```
template<typename Node_CIter, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _Alloc __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc
>::allocator_type
```

`_Alloc` type.

Definition at line 171 of file `trie_policy.hpp`.

#### 5.400.2.4 `size_type`

```
template<typename Node_CIter, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_↵
Itr, _ATraits, _Alloc >::size_type
```

Size type.

Definition at line 174 of file `trie_policy.hpp`.

### 5.400.3 Member Function Documentation

### 5.400.3.1 `operator()`

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
void __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator()
(
    node_iterator node_it,
    node_const_iterator end_nd_it ) const [inline], [protected]
```

Called to update a node's metadata.

Definition at line 139 of file `trie_policy.hpp`.

### 5.400.3.2 `prefix_range()` [1/4]

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
    key_const_reference r_key ) const
```

Finds the const iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 47 of file `trie_policy.hpp`.

### 5.400.3.3 `prefix_range()` [2/4]

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
    key_const_reference r_key )
```

Finds the iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 58 of file `trie_policy.hpp`.

### 5.400.3.4 `prefix_range()` [3/4]

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
    a_const_iterator b,
    a_const_iterator e ) const
```

Finds the const iterator range corresponding to all values whose prefixes match `[b, e)`.

Definition at line 69 of file `trie_policy.hpp`.

5.400.3.5 `prefix_range()` [4/4]

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie\_prefix\_search\_node\_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator, typename trie\_prefix\_search\_node\_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator > \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
    a\_const\_iterator b,
    a\_const\_iterator e )
```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 84 of file `trie_policy.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

5.401 `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >` Struct Template Reference

## Public Types

- enum { **reverse** }
- enum { **min\_e\_val**, **max\_e\_val**, **max\_size** }
- typedef `_Alloc::template rebind< key_type > __rebind_k`
- typedef `detail::__conditional_type< Reverse, typename String::const_reverse_iterator, typename String::const↵_iterator >::__type const\_iterator`
- typedef `std::iterator_traits< const\_iterator >::value_type e\_type`
- typedef `__rebind_k::other::const_reference key_const_reference`
- typedef `String key_type`
- typedef `_Alloc::size_type size_type`

## Static Public Member Functions

- static `const\_iterator begin (key_const_reference)`
- static `size_type e\_pos (e\_type e)`
- static `const\_iterator end (key_const_reference)`

## 5.401.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value↵_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Re↵verse = false, typename _Alloc = std::allocator<char>>
struct \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

## Template Parameters

<i>String</i>	String type.
<i>Min_E_Val</i>	Minimal element value.
<i>Max_E_Val</i>	Maximum element value.
<i>Reverse</i>	Reverse iteration should be used. Default: false.
<i>_Alloc</i>	Allocator type.

Definition at line 74 of file `trie_policy.hpp`.

## 5.401.2 Member Typedef Documentation

5.401.2.1 `const_iterator`

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__  
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__  
numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =  
std::allocator<char>>  
typedef detail::__conditional_type<Reverse, typename String::const_reverse_iterator, typename  
String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_  
E_Val, Reverse, _Alloc >::const_iterator
```

Element `const_iterator` type.

Definition at line 90 of file `trie_policy.hpp`.

5.401.2.2 `e_type`

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__  
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__  
numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =  
std::allocator<char>>  
typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits<  
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type
```

Element type.

Definition at line 93 of file `trie_policy.hpp`.

## 5.401.3 Member Function Documentation



#### 5.401.3.1 begin()

```
template<typename String , typename String::value_type Min_E_Val, typename String::value_type
Max_E_Val, bool Reverse, typename _Alloc >
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (
    key_const_reference r_key ) [inline], [static]
```

Returns a const\_iterator to the first element of key\_const\_reference argument.

Definition at line 57 of file trie\_policy.hpp.

#### 5.401.3.2 e\_pos()

```
template<typename String , typename String::value_type Min_E_Val, typename String::value_type
Max_E_Val, bool Reverse, typename _Alloc >
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (
    e_type e ) [inline], [static]
```

Maps an element to a position.

Definition at line 49 of file trie\_policy.hpp.

#### 5.401.3.3 end()

```
template<typename String , typename String::value_type Min_E_Val, typename String::value_type
Max_E_Val, bool Reverse, typename _Alloc >
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (
    key_const_reference r_key ) [inline], [static]
```

Returns a const\_iterator to the after-last element of key\_const\_reference argument.

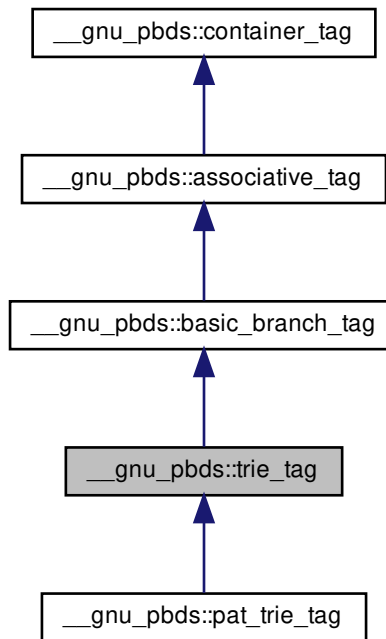
Definition at line 65 of file trie\_policy.hpp.

The documentation for this struct was generated from the following file:

- [trie\\_policy.hpp](#)

## 5.402 \_\_gnu\_pbds::trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::trie\_tag:



### 5.402.1 Detailed Description

Basic trie structure.

Definition at line 162 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.403 \_\_gnu\_pbds::trivial\_iterator\_tag Struct Reference

### 5.403.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of `std::iterators`'s movement abilities.

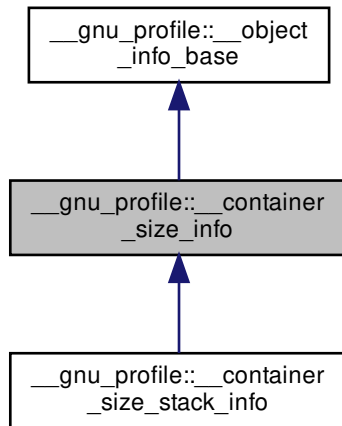
Definition at line 75 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.404 \_\_gnu\_profile::\_\_container\_size\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_container\_size\_info:



### Public Member Functions

- **\_\_container\_size\_info** (\_\_stack\_t \_\_stack)
- **std::string \_\_advice** () const
- void **\_\_destruct** (std::size\_t \_\_num, std::size\_t \_\_inum)
- void **\_\_init** (std::size\_t \_\_num)
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const \_\_container\_size\_info &\_\_o)
- void **\_\_merge** (const \_\_object\_info\_base &\_\_o)
- void **\_\_resize** (std::size\_t \_\_from, std::size\_t \_\_to)
- float **\_\_resize\_cost** (std::size\_t \_\_from, std::size\_t)
- void **\_\_set\_invalid** ()
- \_\_stack\_t **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

### Protected Attributes

- \_\_stack\_t **\_M\_stack**
- bool **\_M\_valid**

## 5.404.1 Detailed Description

A container size instrumentation line in the object table.

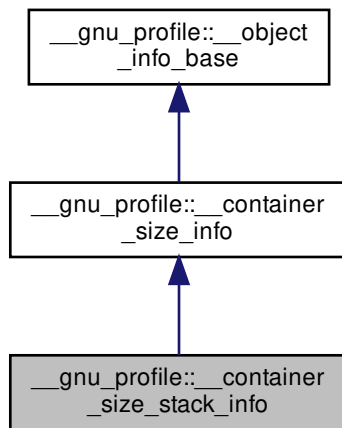
Definition at line 42 of file profiler\_container\_size.h.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

## 5.405 \_\_gnu\_profile::\_\_container\_size\_stack\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_container\_size\_stack\_info:



## Public Member Functions

- **\_\_container\_size\_stack\_info** (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- **std::string \_\_advice** () const
- void **\_\_destruct** (std::size\_t \_\_num, std::size\_t \_\_inum)
- void **\_\_init** (std::size\_t \_\_num)
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- void **\_\_merge** (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- void **\_\_resize** (std::size\_t \_\_from, std::size\_t \_\_to)
- float **\_\_resize\_cost** (std::size\_t \_\_from, std::size\_t)
- void **\_\_set\_invalid** ()
- **\_\_stack\_t \_\_stack** () const
- void **\_\_write** (FILE \* \_\_f) const

### Protected Attributes

- `__stack_t __M_stack`
- `bool __M_valid`

#### 5.405.1 Detailed Description

A container size instrumentation line in the stack table.

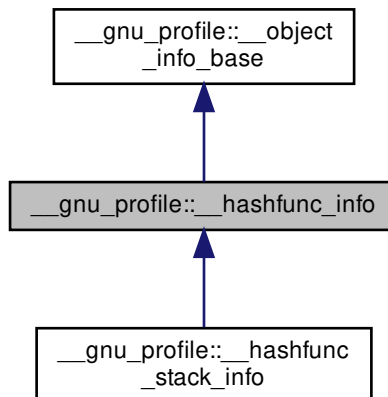
Definition at line 147 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

#### 5.406 `__gnu_profile::__hashfunc_info` Class Reference

Inheritance diagram for `__gnu_profile::__hashfunc_info`:



### Public Member Functions

- `__hashfunc_info (__stack_t __stack)`
- `std::string __advice () const`
- `void __destruct (std::size_t __chain, std::size_t __accesses, std::size_t __hops)`
- `bool __is_valid () const`
- `float __magnitude () const`
- `void __merge (const __hashfunc_info &__o)`
- `void __merge (const __object_info_base &__o)`
- `void __set_invalid ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

## Protected Attributes

- `__stack_t __M_stack`
- `bool __M_valid`

## 5.406.1 Detailed Description

A hash performance instrumentation line in the object table.

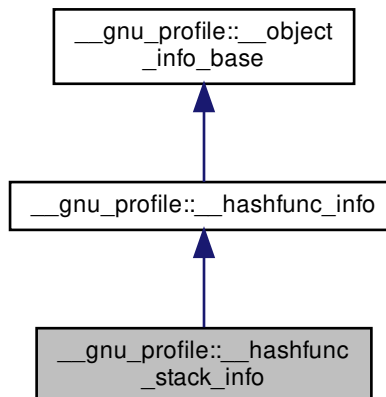
Definition at line 40 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

5.407 `__gnu_profile::__hashfunc_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__hashfunc_stack_info`:



## Public Member Functions

- `__hashfunc_stack_info` (const [\\_\\_hashfunc\\_info](#) &\_\_o)
- `std::string __advice` () const
- `void __destruct` (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const [\\_\\_hashfunc\\_info](#) &\_\_o)
- `void __merge` (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- `void __set_invalid` ()
- `__stack_t __stack` () const
- `void __write` (FILE \* \_\_f) const

### Protected Attributes

- `__stack_t __M_stack`
- `bool __M_valid`

### 5.407.1 Detailed Description

A hash performance instrumentation line in the stack table.

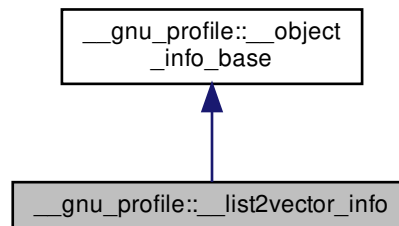
Definition at line 86 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

### 5.408 `__gnu_profile::__list2vector_info` Class Reference

Inheritance diagram for `__gnu_profile::__list2vector_info`:



### Public Member Functions

- `__list2vector_info (__stack_t __stack)`
- `std::string __advice () const`
- `bool __is_valid () const`
- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __list2vector_info &__o)`
- `void __merge (const __object_info_base &__o)`
- `void __opr_insert (std::size_t __shift, std::size_t __size)`
- `void __opr_iterate (int __num)`
- `std::size_t __resize ()`
- `void __resize (std::size_t __from, std::size_t)`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

## Protected Attributes

- \_\_stack\_t **M\_stack**
- bool **M\_valid**

## 5.408.1 Detailed Description

A list-to-vector instrumentation line in the object table.

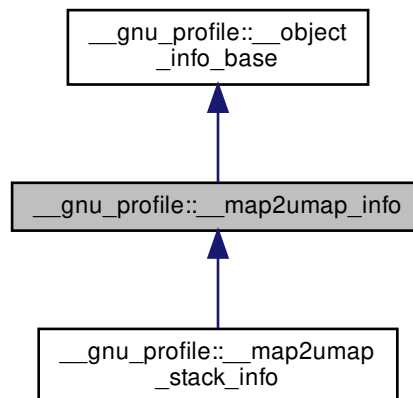
Definition at line 42 of file profiler\_list\_to\_vector.h.

The documentation for this class was generated from the following file:

- [profiler\\_list\\_to\\_vector.h](#)

## 5.409 \_\_gnu\_profile::\_\_map2umap\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_map2umap\_info:



## Public Member Functions

- **\_\_map2umap\_info** (\_\_stack\_t \_\_stack)
- [std::string](#) **\_\_advice** () const
- bool **\_\_is\_valid** () const
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_map2umap\\_info](#) &\_\_o)
- void **\_\_merge** (const [\\_\\_object\\_info\\_base](#) &\_\_o)



- void **\_\_record\_erase** (std::size\_t \_\_size, std::size\_t \_\_count)
- void **\_\_record\_find** (std::size\_t \_\_size)
- void **\_\_record\_insert** (std::size\_t \_\_size, std::size\_t \_\_count)
- void **\_\_record\_iterate** (int \_\_count)
- void **\_\_set\_invalid** ()
- void **\_\_set\_iterate\_costs** ()
- \_\_stack\_t **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- \_\_stack\_t **\_M\_stack**
- bool **\_M\_valid**

#### 5.409.1 Detailed Description

A map-to-unordered\_map instrumentation line in the object table.

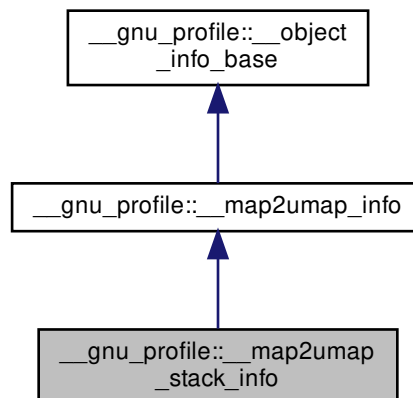
Definition at line 66 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

#### 5.410 \_\_gnu\_profile::\_\_map2umap\_stack\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_map2umap\_stack\_info:



## Public Member Functions

- [\\_\\_map2umap\\_stack\\_info](#) (const [\\_\\_map2umap\\_info](#) &\_\_o)
- [std::string \\_\\_advice](#) () const
- bool [\\_\\_is\\_valid](#) () const
- float [\\_\\_magnitude](#) () const
- void [\\_\\_merge](#) (const [\\_\\_map2umap\\_info](#) &\_\_o)
- void [\\_\\_merge](#) (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- void [\\_\\_record\\_erase](#) (std::size\_t \_\_size, std::size\_t \_\_count)
- void [\\_\\_record\\_find](#) (std::size\_t \_\_size)
- void [\\_\\_record\\_insert](#) (std::size\_t \_\_size, std::size\_t \_\_count)
- void [\\_\\_record\\_iterate](#) (int \_\_count)
- void [\\_\\_set\\_invalid](#) ()
- void [\\_\\_set\\_iterate\\_costs](#) ()
- [\\_\\_stack\\_t \\_\\_stack](#) () const
- void [\\_\\_write](#) (FILE \*\_\_f) const

## Protected Attributes

- [\\_\\_stack\\_t \\_M\\_stack](#)
- bool [\\_M\\_valid](#)

## 5.410.1 Detailed Description

A map-to-unordered\_map instrumentation line in the stack table.

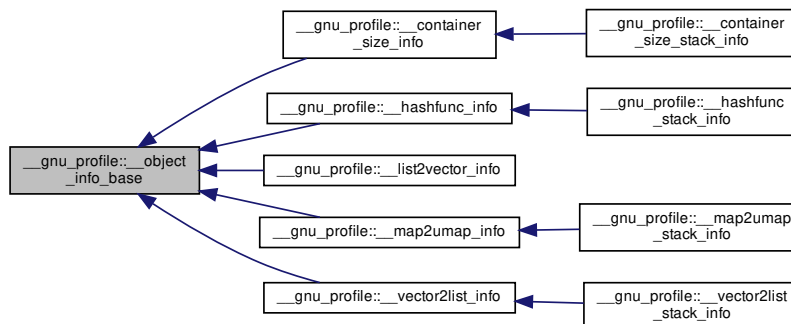
Definition at line 164 of file profiler\_map\_to\_unordered\_map.h.

The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.411 \_\_gnu\_profile::\_\_object\_info\_base Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_object\_info\_base:



#### Public Member Functions

- **\_\_object\_info\_base** (\_\_stack\_t \_\_stack)
- bool **\_\_is\_valid** () const
- void **\_\_merge** (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- void **\_\_set\_invalid** ()
- **\_\_stack\_t \_\_stack** () const

#### Protected Attributes

- **\_\_stack\_t M\_stack**
- bool **M\_valid**

##### 5.411.1 Detailed Description

Base class for a line in the object table.

Definition at line 127 of file profiler\_node.h.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

##### 5.412 \_\_gnu\_profile::\_\_reentrance\_guard Struct Reference

#### Static Public Member Functions

- static bool **\_\_get\_in** ()
- static bool & **\_\_inside** ()

##### 5.412.1 Detailed Description

Reentrance guard.

Mechanism to protect all \_\_gnu\_profile operations against recursion, multithreaded and exception reentrance.

Definition at line 58 of file profiler.h.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

## 5.413 `__gnu_profile::__stack_hash` Class Reference

### Public Member Functions

- `std::size_t operator() (__stack_t __s) const`
- `bool operator() (__stack_t __stack1, __stack_t __stack2) const`

#### 5.413.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 93 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.414 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

### Public Member Functions

- `__object_info * __add_object (__stack_t __stack)`
- `void __collect_warnings (__warning_vector_t &__warnings)`
- `void __free ()`
- `void __retire_object (__object_info * __info)`
- `void __write (FILE * __f)`

### Protected Attributes

- `const char * __id`

#### 5.414.1 Detailed Description

```
template<typename __object_info, typename __stack_info>
class __gnu_profile::__trace_base< __object_info, __stack_info >
```

Base class for all trace producers.

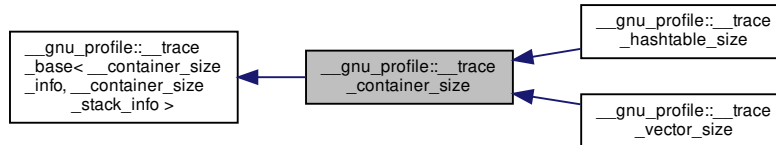
Definition at line 190 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler\\_trace.h](#)

## 5.415 `__gnu_profile::__trace_container_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_container_size`:



### Public Member Functions

- `__container_size_info * __add_object (__stack_t __stack)`
- `void __collect_warnings (__warning_vector_t &__warnings)`
- `void __destruct (__container_size_info * __obj_info, std::size_t __num, std::size_t __inum)`
- `void __free ()`
- `__container_size_info * __insert (__stack_t __stack, std::size_t __num)`
- `void __retire_object (__container_size_info * __info)`
- `void __write (FILE * __f)`

### Protected Attributes

- `const char * __id`

### 5.415.1 Detailed Description

Container size instrumentation trace producer.

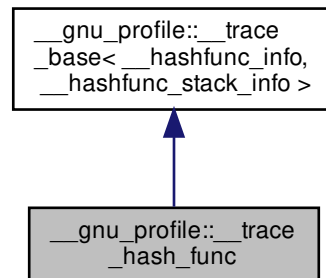
Definition at line 156 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

## 5.416 \_\_gnu\_profile::\_\_trace\_hash\_func Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_hash\_func:



## Public Member Functions

- [\\_\\_hashfunc\\_info](#) \* **add\_object** (\_\_stack\_t \_\_stack)
- void **collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- void **destruct** (\_\_hashfunc\_info \*\_\_obj\_info, std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- void **free** ()
- void **retire\_object** (\_\_hashfunc\_info \*\_\_info)
- void **write** (FILE \*\_\_f)

## Protected Attributes

- const char \* **\_\_id**

## 5.416.1 Detailed Description

Hash performance instrumentation producer.

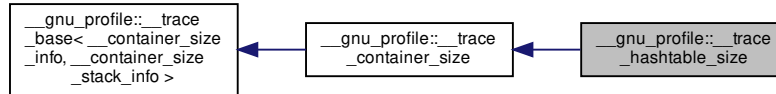
Definition at line 95 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

## 5.417 \_\_gnu\_profile::\_\_trace\_hashtable\_size Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_hashtable\_size:



### Public Member Functions

- [\\_\\_container\\_size\\_info](#) \* **\_\_add\_object** (\_\_stack\_t \_\_stack)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- void **\_\_destruct** ([\\_\\_container\\_size\\_info](#) \* \_\_obj\_info, std::size\_t \_\_num, std::size\_t \_\_inum)
- void **\_\_free** ()
- [\\_\\_container\\_size\\_info](#) \* **\_\_insert** (\_\_stack\_t \_\_stack, std::size\_t \_\_num)
- void **\_\_retire\_object** ([\\_\\_container\\_size\\_info](#) \* \_\_info)
- void **\_\_write** (FILE \* \_\_f)

### Protected Attributes

- const char \* **\_\_id**

### 5.417.1 Detailed Description

Hashtable size instrumentation trace producer.

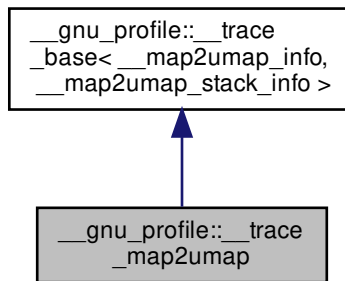
Definition at line 42 of file profiler\_hashtable\_size.h.

The documentation for this class was generated from the following file:

- [profiler\\_hashtable\\_size.h](#)

5.418 `__gnu_profile::__trace_map2umap` Class Reference

Inheritance diagram for `__gnu_profile::__trace_map2umap`:



## Public Member Functions

- `__map2umap_info * __add_object (__stack_t __stack)`
- `void __collect_warnings (__warning_vector_t & __warnings)`
- `void __destruct (__map2umap_info * __obj_info)`
- `void __free ()`
- `void __retire_object (__map2umap_info * __info)`
- `void __write (FILE * __f)`

## Protected Attributes

- `const char * __id`

## 5.418.1 Detailed Description

Map-to-unordered\_map instrumentation producer.

Definition at line 173 of file `profiler_map_to_unordered_map.h`.

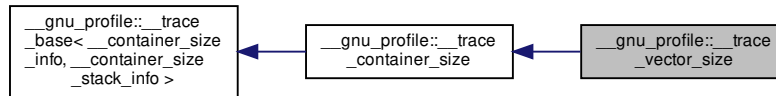
The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)



## 5.419 \_\_gnu\_profile::\_\_trace\_vector\_size Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_vector\_size:



### Public Member Functions

- [\\_\\_container\\_size\\_info](#) \* **\_\_add\_object** (\_\_stack\_t \_\_stack)
- void **\_\_collect\_warnings** (\_\_warning\_vector\_t &\_\_warnings)
- void **\_\_destruct** (\_\_container\_size\_info \* \_\_obj\_info, std::size\_t \_\_num, std::size\_t \_\_inum)
- void **\_\_free** ()
- [\\_\\_container\\_size\\_info](#) \* **\_\_insert** (\_\_stack\_t \_\_stack, std::size\_t \_\_num)
- void **\_\_retire\_object** (\_\_container\_size\_info \* \_\_info)
- void **\_\_write** (FILE \* \_\_f)

### Protected Attributes

- const char \* **\_\_id**

### 5.419.1 Detailed Description

Hashtable size instrumentation trace producer.

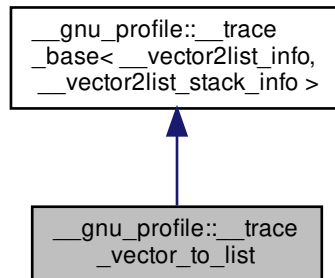
Definition at line 42 of file profiler\_vector\_size.h.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_size.h](#)

5.420 `__gnu_profile::__trace_vector_to_list` Class Reference

Inheritance diagram for `__gnu_profile::__trace_vector_to_list`:



## Public Member Functions

- `__vector2list_info * __add_object (__stack_t __stack)`
- `void __collect_warnings (__warning_vector_t & __warnings)`
- `void __destruct (__vector2list_info * __obj_info)`
- `void __free ()`
- `float __list_cost (std::size_t __shift, std::size_t __iterate, std::size_t __resize)`
- `void __retire_object (__vector2list_info * __info)`
- `float __vector_cost (std::size_t __shift, std::size_t __iterate, std::size_t __resize)`
- `void __write (FILE * __f)`

## Protected Attributes

- `const char * __id`

## 5.420.1 Detailed Description

Vector-to-list instrumentation producer.

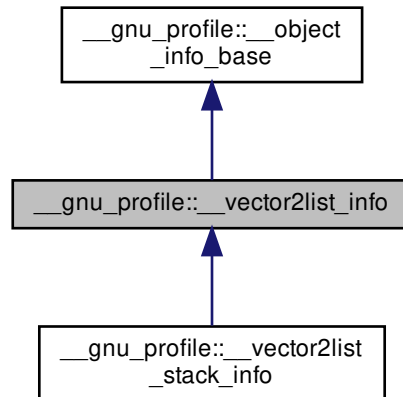
Definition at line 131 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.421 \_\_gnu\_profile::\_\_vector2list\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_vector2list\_info:



### Public Member Functions

- **\_\_vector2list\_info** (\_\_stack\_t \_\_stack)
- [std::string \\_\\_advice](#) () const
- bool **\_\_is\_valid** () const
- std::size\_t **\_\_iterate** ()
- float **\_\_list\_cost** ()
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_vector2list\\_info](#) &\_\_o)
- void **\_\_merge** (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- void **\_\_opr\_insert** (std::size\_t \_\_pos, std::size\_t \_\_num)
- void **\_\_opr\_iterate** (int \_\_num)
- std::size\_t **\_\_resize** ()
- void **\_\_resize** (std::size\_t \_\_from, std::size\_t)
- void **\_\_set\_invalid** ()
- void **\_\_set\_list\_cost** (float \_\_lc)
- void **\_\_set\_vector\_cost** (float \_\_vc)
- std::size\_t **\_\_shift\_count** ()
- \_\_stack\_t **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

### Protected Attributes

- \_\_stack\_t **M\_stack**
- bool **M\_valid**

## 5.421.1 Detailed Description

A vector-to-list instrumentation line in the object table.

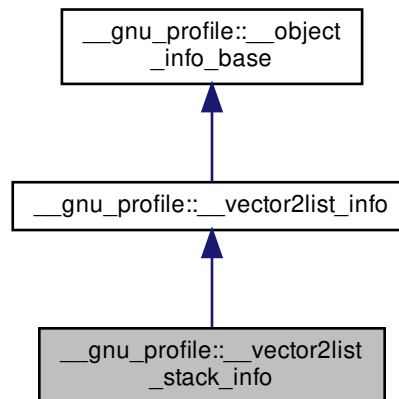
Definition at line 40 of file profiler\_vector\_to\_list.h.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

## 5.422 \_\_gnu\_profile::\_\_vector2list\_stack\_info Class Reference

Inheritance diagram for \_\_gnu\_profile::\_\_vector2list\_stack\_info:



## Public Member Functions

- **\_\_vector2list\_stack\_info** (const [\\_\\_vector2list\\_info](#) &\_\_o)
- [std::string](#) **\_\_advice** () const
- bool **\_\_is\_valid** () const
- std::size\_t **\_\_iterate** ()
- float **\_\_list\_cost** ()
- float **\_\_magnitude** () const
- void **\_\_merge** (const [\\_\\_vector2list\\_info](#) &\_\_o)
- void **\_\_merge** (const [\\_\\_object\\_info\\_base](#) &\_\_o)
- void **\_\_opr\_insert** (std::size\_t \_\_pos, std::size\_t \_\_num)
- void **\_\_opr\_iterate** (int \_\_num)
- std::size\_t **\_\_resize** ()
- void **\_\_resize** (std::size\_t \_\_from, std::size\_t)
- void **\_\_set\_invalid** ()
- void **\_\_set\_list\_cost** (float \_\_lc)
- void **\_\_set\_vector\_cost** (float \_\_vc)
- std::size\_t **\_\_shift\_count** ()
- [\\_\\_stack\\_t](#) **\_\_stack** () const
- void **\_\_write** (FILE \* \_\_f) const

#### Protected Attributes

- `__stack_t` **`_M_stack`**
- `bool` **`_M_valid`**

##### 5.422.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 121 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

#### 5.423 `__gnu_profile::__warning_data` Struct Reference

##### Public Member Functions

- **`__warning_data`** (`float` `__m`, `__stack_t` `__c`, `const char *``__id`, `const` [std::string](#) &`__msg`)
- `bool` **`operator<`** (`const` [\\_\\_warning\\_data](#) &`__other`) `const`

##### Public Attributes

- `__stack_t` **`__context`**
- `float` **`__magnitude`**
- `const char *` **`__warning_id`**
- [std::string](#) **`__warning_message`**

##### 5.423.1 Detailed Description

Representation of a warning.

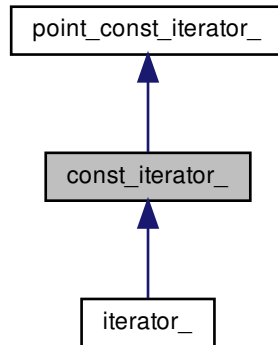
Definition at line 73 of file `profiler_trace.h`.

The documentation for this struct was generated from the following file:

- [profiler\\_trace.h](#)

5.424 `const_iterator_` Class Reference

Inheritance diagram for `const_iterator_`:



## Public Types

- typedef `const_pointer_` [const\\_pointer](#)
- typedef `const_reference_` [const\\_reference](#)
- typedef `_Alloc::difference_type` [difference\\_type](#)
- typedef `std::forward_iterator_tag` [iterator\\_category](#)
- typedef `pointer_` [pointer](#)
- typedef `reference_` [reference](#)
- typedef `value_type_` [value\\_type](#)

## Public Member Functions

- [const\\_iterator\\_](#) ()
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [const\\_reference](#) [operator\\*](#) () const
- [const\\_iterator\\_](#) & [operator++](#) ()
- [const\\_iterator\\_](#) [operator++](#) (int)
- [const\\_pointer](#) [operator->](#) () const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

## Protected Types

- typedef [point\\_const\\_iterator\\_](#) **base\_type**

#### Protected Member Functions

- [const\\_iterator\\_](#) (const\_pointer\_ p\_value, PB\_DS\_GEN\_POS pos, const PB\_DS\_CLASS\_C\_DEC \*p\_tbl)

#### Protected Attributes

- const PB\_DS\_CLASS\_C\_DEC \* [m\\_p\\_tbl](#)
- [const\\_pointer](#) [m\\_p\\_value](#)
- PB\_DS\_GEN\_POS [m\\_pos](#)

#### Friends

- class **PB\_DS\_CLASS\_C\_DEC**

#### 5.424.1 Detailed Description

Const range-type iterator.

Definition at line 43 of file unordered\_iterator/const\_iterator.hpp.

#### 5.424.2 Member Typedef Documentation

##### 5.424.2.1 const\_pointer

```
typedef const_pointer_ const\_iterator\_::const\_pointer
```

Iterator's const pointer type.

Definition at line 60 of file unordered\_iterator/const\_iterator.hpp.

##### 5.424.2.2 const\_reference

```
typedef const_reference_ const\_iterator\_::const\_reference
```

Iterator's const reference type.

Definition at line 66 of file unordered\_iterator/const\_iterator.hpp.

#### 5.424.2.3 difference\_type

```
typedef _Alloc::difference_type const_iterator::difference_type
```

Difference type.

Definition at line 51 of file unordered\_iterator/const\_iterator.hpp.

#### 5.424.2.4 iterator\_category

```
typedef std::forward_iterator_tag const_iterator::iterator_category
```

Category.

Definition at line 48 of file unordered\_iterator/const\_iterator.hpp.

#### 5.424.2.5 pointer

```
typedef pointer_ const_iterator::pointer
```

Iterator's pointer type.

Definition at line 57 of file unordered\_iterator/const\_iterator.hpp.

#### 5.424.2.6 reference

```
typedef reference_ const_iterator::reference
```

Iterator's reference type.

Definition at line 63 of file unordered\_iterator/const\_iterator.hpp.

#### 5.424.2.7 value\_type

```
typedef value_type_ const_iterator::value_type
```

Iterator's value type.

Definition at line 54 of file unordered\_iterator/const\_iterator.hpp.



### 5.424.3 Constructor & Destructor Documentation

#### 5.424.3.1 `const_iterator_()` [1/2]

```
const_iterator_::const_iterator_ ( ) [inline]
```

Default constructor.

Definition at line 69 of file `unordered_iterator/const_iterator.hpp`.

#### 5.424.3.2 `const_iterator_()` [2/2]

```
const_iterator_::const_iterator_ (
    const_pointer_ p_value,
    PB_DS_GEN_POS pos,
    const PB_DS_CLASS_C_DEC * p_tbl ) [inline], [protected]
```

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a `find()` of a table).

Definition at line 97 of file `unordered_iterator/const_iterator.hpp`.

### 5.424.4 Member Function Documentation

#### 5.424.4.1 `operator!=( )` [1/2]

```
bool point_const_iterator_::operator!= (
    const point_iterator_ & other ) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 118 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.424.4.2 `operator!=( )` [2/2]

```
bool point_const_iterator_::operator!= (
    const point_const_iterator_ & other ) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 123 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.424.4.3 operator\*()

```
const_reference point_const_iterator_::operator* ( ) const [inline], [inherited]
```

Access.

Definition at line 100 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.424.4.4 operator++() [1/2]

```
const_iterator_ & const_iterator_::operator++ ( ) [inline]
```

Increments.

Definition at line 74 of file unordered\_iterator/const\_iterator.hpp.

References `m_p_tbl`.

#### 5.424.4.5 operator++() [2/2]

```
const_iterator_ const_iterator_::operator++ (
    int ) [inline]
```

Increments.

Definition at line 82 of file unordered\_iterator/const\_iterator.hpp.

References `m_p_tbl`.

#### 5.424.4.6 operator->()

```
const_pointer point_const_iterator_::operator-> ( ) const [inline], [inherited]
```

Access.

Definition at line 92 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.424.4.7 operator==( ) [1/2]

```
bool point_const_iterator_::operator==(
    const point_iterator_ & other ) const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 108 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.424.4.8 operator==( ) [2/2]

```
bool point_const_iterator_::operator==(
    const point\_const\_iterator\_ & other ) const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 113 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.424.5 Member Data Documentation

##### 5.424.5.1 m\_p\_tbl

```
const PB_DS_CLASS_C_DEC* const_iterator_::m_p_tbl [protected]
```

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file unordered\_iterator/const\_iterator.hpp.

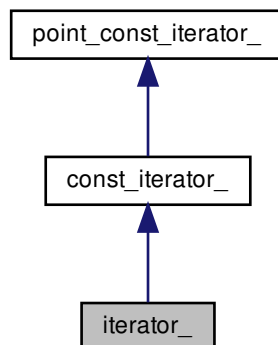
Referenced by operator++(), and iterator\_::operator++().

The documentation for this class was generated from the following file:

- [unordered\\_iterator/const\\_iterator.hpp](#)

#### 5.425 iterator\_ Class Reference

Inheritance diagram for iterator\_:



### Public Types

- typedef const\_pointer\_ [const\\_pointer](#)
- typedef const\_reference\_ [const\\_reference](#)
- typedef \_Alloc::difference\_type [difference\\_type](#)
- typedef [std::forward\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef pointer\_ [pointer](#)
- typedef reference\_ [reference](#)
- typedef value\_type\_ [value\\_type](#)

### Public Member Functions

- [iterator\\_](#) ()
- [operator const point\\_iterator\\_](#) () const
- [operator point\\_iterator\\_](#) ()
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [reference operator\\*](#) () const
- [iterator\\_ & operator++](#) ()
- [iterator\\_ operator++](#) (int)
- [pointer operator->](#) () const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

### Protected Types

- typedef [const\\_iterator\\_](#) **base\_type**

### Protected Member Functions

- [iterator\\_](#) ([pointer](#) p\_value, PB\_DS\_GEN\_POS pos, PB\_DS\_CLASS\_C\_DEC \*p\_tbl)

### Protected Attributes

- const PB\_DS\_CLASS\_C\_DEC \* [m\\_p\\_tbl](#)
- [const\\_pointer](#) [m\\_p\\_value](#)
- PB\_DS\_GEN\_POS [m\\_pos](#)

### Friends

- class **PB\_DS\_CLASS\_C\_DEC**

#### 5.425.1 Detailed Description

Range-type iterator.

Definition at line 43 of file iterator.hpp.

## 5.425.2 Member Typedef Documentation

### 5.425.2.1 `const_pointer`

```
typedef const_pointer_ iterator_::const_pointer
```

Iterator's const pointer type.

Definition at line 60 of file iterator.hpp.

### 5.425.2.2 `const_reference`

```
typedef const_reference_ iterator_::const_reference
```

Iterator's const reference type.

Definition at line 66 of file iterator.hpp.

### 5.425.2.3 `difference_type`

```
typedef _Alloc::difference_type iterator_::difference_type
```

Difference type.

Definition at line 51 of file iterator.hpp.

### 5.425.2.4 `iterator_category`

```
typedef std::forward_iterator_tag iterator_::iterator_category
```

Category.

Definition at line 48 of file iterator.hpp.

### 5.425.2.5 `pointer`

```
typedef pointer_ iterator_::pointer
```

Iterator's pointer type.

Definition at line 57 of file iterator.hpp.

#### 5.425.2.6 reference

```
typedef reference_ iterator_::reference
```

Iterator's reference type.

Definition at line 63 of file iterator.hpp.

#### 5.425.2.7 value\_type

```
typedef value_type_ iterator_::value_type
```

Iterator's value type.

Definition at line 54 of file iterator.hpp.

### 5.425.3 Constructor & Destructor Documentation

#### 5.425.3.1 iterator\_() [1/2]

```
iterator_::iterator_ ( ) [inline]
```

Default constructor.

Definition at line 70 of file iterator.hpp.

#### 5.425.3.2 iterator\_() [2/2]

```
iterator_::iterator_ (
    pointer p_value,
    PB_DS_GEN_POS pos,
    PB_DS_CLASS_C_DEC * p_tbl ) [inline], [protected]
```

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a find() of a table.

Definition at line 125 of file iterator.hpp.

### 5.425.4 Member Function Documentation

#### 5.425.4.1 operator const point\_iterator\_()

```
iterator_::operator const point_iterator_ ( ) const [inline]
```

Conversion to a point-type iterator.

Definition at line 80 of file iterator.hpp.

#### 5.425.4.2 operator point\_iterator\_()

```
iterator_::operator point_iterator_ ( ) [inline]
```

Conversion to a point-type iterator.

Definition at line 75 of file iterator.hpp.

#### 5.425.4.3 operator!=( ) [1/2]

```
bool point_const_iterator_::operator!= (
    const point_iterator_ & other ) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.425.4.4 operator!=( ) [2/2]

```
bool point_const_iterator_::operator!= (
    const point_const_iterator_ & other ) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.425.4.5 operator\*()

```
reference iterator_::operator* ( ) const [inline]
```

Access.

Definition at line 93 of file iterator.hpp.

#### 5.425.4.6 operator++() [1/2]

```
iterator_& iterator_::operator++ ( ) [inline]
```

Increments.

Definition at line 101 of file iterator.hpp.

References `const_iterator_::m_p_tbl`.

#### 5.425.4.7 operator++() [2/2]

```
iterator_ iterator_::operator++ (
    int ) [inline]
```

Increments.

Definition at line 109 of file iterator.hpp.

References `const_iterator_::m_p_tbl`.

#### 5.425.4.8 operator->()

```
pointer iterator_::operator-> ( ) const [inline]
```

Access.

Definition at line 85 of file iterator.hpp.

#### 5.425.4.9 operator==( ) [1/2]

```
bool point_const_iterator_::operator== (
    const point_iterator_ & other ) const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 108 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.425.4.10 operator==( ) [2/2]

```
bool point_const_iterator_::operator== (
    const point_const_iterator_ & other ) const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 113 of file unordered\_iterator/point\_const\_iterator.hpp.



### 5.425.5 Member Data Documentation

#### 5.425.5.1 m\_p\_tbl

```
const PB_DS_CLASS_C_DEC* const_iterator_::m_p_tbl [protected], [inherited]
```

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file unordered\_iterator/const\_iterator.hpp.

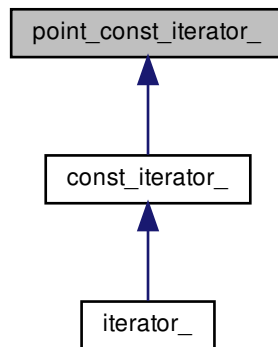
Referenced by `const_iterator_::operator++()`, and `operator++()`.

The documentation for this class was generated from the following file:

- [iterator.hpp](#)

### 5.426 point\_const\_iterator\_ Class Reference

Inheritance diagram for `point_const_iterator_`:



#### Public Types

- typedef const\_pointer\_ [const\\_pointer](#)
- typedef const\_reference\_ [const\\_reference](#)
- typedef trivial\_iterator\_difference\_type [difference\\_type](#)
- typedef trivial\_iterator\_tag [iterator\\_category](#)
- typedef pointer\_ [pointer](#)
- typedef reference\_ [reference](#)
- typedef value\_type\_ [value\\_type](#)

## Public Member Functions

- **point\_const\_iterator\_** (const\_pointer p\_value)
- **point\_const\_iterator\_** ()
- **point\_const\_iterator\_** (const **point\_const\_iterator\_** &other)
- **point\_const\_iterator\_** (const **point\_iterator\_** &other)
- bool **operator!=** (const **point\_iterator\_** &other) const
- bool **operator!=** (const **point\_const\_iterator\_** &other) const
- **const\_reference operator\*** () const
- **const\_pointer operator->** () const
- bool **operator==** (const **point\_iterator\_** &other) const
- bool **operator==** (const **point\_const\_iterator\_** &other) const

## Protected Attributes

- **const\_pointer m\_p\_value**

## Friends

- class **PB\_DS\_CLASS\_C\_DEC**
- class **point\_iterator\_**

## 5.426.1 Detailed Description

Const point-type iterator.

Definition at line 45 of file unordered\_iterator/point\_const\_iterator.hpp.

## 5.426.2 Member Typedef Documentation

## 5.426.2.1 const\_pointer

```
typedef const_pointer_ point_const_iterator_::const_pointer
```

Iterator's const pointer type.

Definition at line 61 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.426.2.2 `const_reference`

```
typedef const_reference_ point_const_iterator_::const_reference
```

Iterator's const reference type.

Definition at line 67 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.426.2.3 `difference_type`

```
typedef trivial_iterator_difference_type point_const_iterator_::difference_type
```

Difference type.

Definition at line 52 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.426.2.4 `iterator_category`

```
typedef trivial_iterator_tag point_const_iterator_::iterator_category
```

Category.

Definition at line 49 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.426.2.5 `pointer`

```
typedef pointer_ point_const_iterator_::pointer
```

Iterator's pointer type.

Definition at line 58 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.426.2.6 `reference`

```
typedef reference_ point_const_iterator_::reference
```

Iterator's reference type.

Definition at line 64 of file `unordered_iterator/point_const_iterator.hpp`.

#### 5.426.2.7 value\_type

```
typedef value_type_ point_const_iterator_::value_type
```

Iterator's value type.

Definition at line 55 of file unordered\_iterator/point\_const\_iterator.hpp.

### 5.426.3 Constructor & Destructor Documentation

#### 5.426.3.1 point\_const\_iterator\_() [1/3]

```
point_const_iterator_::point_const_iterator_ ( ) [inline]
```

Default constructor.

Definition at line 75 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.426.3.2 point\_const\_iterator\_() [2/3]

```
point_const_iterator_::point_const_iterator_ (
    const point_const_iterator_ & other ) [inline]
```

Copy constructor.

Definition at line 80 of file unordered\_iterator/point\_const\_iterator.hpp.

#### 5.426.3.3 point\_const\_iterator\_() [3/3]

```
point_const_iterator_::point_const_iterator_ (
    const point_iterator_ & other ) [inline]
```

Copy constructor.

Definition at line 86 of file unordered\_iterator/point\_const\_iterator.hpp.

### 5.426.4 Member Function Documentation

**5.426.4.1 operator!=()** [1/2]

```
bool point_const_iterator_::operator!= (
    const point\_iterator\_ & other ) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 118 of file `unordered_iterator/point_const_iterator.hpp`.

**5.426.4.2 operator!=()** [2/2]

```
bool point_const_iterator_::operator!= (
    const point\_const\_iterator\_ & other ) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 123 of file `unordered_iterator/point_const_iterator.hpp`.

**5.426.4.3 operator\*()**

```
const\_reference point_const_iterator_::operator* ( ) const [inline]
```

Access.

Definition at line 100 of file `unordered_iterator/point_const_iterator.hpp`.

**5.426.4.4 operator->()**

```
const\_pointer point_const_iterator_::operator-> ( ) const [inline]
```

Access.

Definition at line 92 of file `unordered_iterator/point_const_iterator.hpp`.

**5.426.4.5 operator==()** [1/2]

```
bool point_const_iterator_::operator== (
    const point\_iterator\_ & other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 108 of file `unordered_iterator/point_const_iterator.hpp`.

## 5.426.4.6 operator==( ) [ 2 / 2 ]

```
bool point_const_iterator_::operator== (
    const point_const_iterator_ & other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 113 of file unordered\_iterator/point\_const\_iterator.hpp.

The documentation for this class was generated from the following file:

- [unordered\\_iterator/point\\_const\\_iterator.hpp](#)

## 5.427 point\_iterator\_ Class Reference

## Public Types

- typedef const\_pointer\_ [const\\_pointer](#)
- typedef const\_reference\_ [const\\_reference](#)
- typedef trivial\_iterator\_difference\_type [difference\\_type](#)
- typedef trivial\_iterator\_tag [iterator\\_category](#)
- typedef pointer\_ [pointer](#)
- typedef reference\_ [reference](#)
- typedef value\_type\_ [value\\_type](#)

## Public Member Functions

- [point\\_iterator\\_](#) ( )
- [point\\_iterator\\_](#) (const [point\\_iterator\\_](#) &other)
- **point\_iterator\_** ([pointer](#) p\_value)
- bool [operator!=](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator!=](#) (const [point\\_const\\_iterator\\_](#) &other) const
- [reference operator\\*](#) ( ) const
- [pointer operator->](#) ( ) const
- bool [operator==](#) (const [point\\_iterator\\_](#) &other) const
- bool [operator==](#) (const [point\\_const\\_iterator\\_](#) &other) const

## Protected Attributes

- [pointer](#) m\_p\_value

## Friends

- class **PB\_DS\_CLASS\_C\_DEC**
- class [point\\_const\\_iterator\\_](#)

### 5.427.1 Detailed Description

Find type iterator.

Definition at line 43 of file point\_iterator.hpp.

### 5.427.2 Member Typedef Documentation

#### 5.427.2.1 const\_pointer

```
typedef const_pointer_ point_iterator_::const_pointer
```

Iterator's const pointer type.

Definition at line 59 of file point\_iterator.hpp.

#### 5.427.2.2 const\_reference

```
typedef const_reference_ point_iterator_::const_reference
```

Iterator's const reference type.

Definition at line 65 of file point\_iterator.hpp.

#### 5.427.2.3 difference\_type

```
typedef trivial_iterator_difference_type point_iterator_::difference_type
```

Difference type.

Definition at line 50 of file point\_iterator.hpp.

#### 5.427.2.4 iterator\_category

```
typedef trivial_iterator_tag point_iterator_::iterator_category
```

Category.

Definition at line 47 of file point\_iterator.hpp.

#### 5.427.2.5 pointer

```
typedef pointer_ point_iterator_::pointer
```

Iterator's pointer type.

Definition at line 56 of file point\_iterator.hpp.

#### 5.427.2.6 reference

```
typedef reference_ point_iterator_::reference
```

Iterator's reference type.

Definition at line 62 of file point\_iterator.hpp.

#### 5.427.2.7 value\_type

```
typedef value_type_ point_iterator_::value_type
```

Iterator's value type.

Definition at line 53 of file point\_iterator.hpp.

### 5.427.3 Constructor & Destructor Documentation

#### 5.427.3.1 point\_iterator\_() [1/2]

```
point_iterator_::point_iterator_ ( ) [inline]
```

Default constructor.

Definition at line 69 of file point\_iterator.hpp.

#### 5.427.3.2 point\_iterator\_() [2/2]

```
point_iterator_::point_iterator_ (
    const point_iterator_ & other ) [inline]
```

Copy constructor.

Definition at line 75 of file point\_iterator.hpp.



## 5.427.4 Member Function Documentation

### 5.427.4.1 `operator!=()` [1/2]

```
bool point_iterator_::operator!= (
    const point\_iterator\_ & other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 107 of file `point_iterator.hpp`.

### 5.427.4.2 `operator!=()` [2/2]

```
bool point_iterator_::operator!= (
    const point\_const\_iterator\_ & other ) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 112 of file `point_iterator.hpp`.

### 5.427.4.3 `operator*()`

```
reference point_iterator_::operator* ( ) const [inline]
```

Access.

Definition at line 89 of file `point_iterator.hpp`.

### 5.427.4.4 `operator->()`

```
pointer point_iterator_::operator-> ( ) const [inline]
```

Access.

Definition at line 81 of file `point_iterator.hpp`.

## 5.427.4.5 operator==( ) [1/2]

```
bool point_iterator_::operator==(
    const point_iterator_ & other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 97 of file point\_iterator.hpp.

## 5.427.4.6 operator==( ) [2/2]

```
bool point_iterator_::operator==(
    const point_const_iterator_ & other ) const [inline]
```

Compares content to a different iterator object.

Definition at line 102 of file point\_iterator.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterator.hpp](#)

## 5.428 std::\_\_add\_pointer\_helper&lt;\_Tp, bool &gt; Struct Template Reference

## Public Types

- `typedef _Tp type`

## 5.428.1 Detailed Description

```
template<typename _Tp, bool = __or_<__is_referenceable<_Tp>, is_void<_Tp>>::value>
struct std::__add_pointer_helper<_Tp, bool >
```

add\_pointer

Definition at line 1749 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.429 std::\_\_allocated\_ptr<\_Alloc> Struct Template Reference

### Public Types

- using **pointer** = typename [allocator\\_traits](#)<\_Alloc>::pointer
- using **value\_type** = typename [allocator\\_traits](#)<\_Alloc>::value\_type

### Public Member Functions

- [\\_\\_allocated\\_ptr](#) (\_Alloc &\_\_a, pointer \_\_ptr) noexcept
- template<typename \_Ptr, typename \_Req = \_Require<is\_same<\_Ptr, value\_type\*>>>>  
[\\_\\_allocated\\_ptr](#) (\_Alloc &\_\_a, \_Ptr \_\_ptr)
- [\\_\\_allocated\\_ptr](#) ([\\_\\_allocated\\_ptr](#) &&\_\_gd) noexcept
- [~\\_\\_allocated\\_ptr](#) ()
- value\_type \* [get](#) ()
- [\\_\\_allocated\\_ptr](#) & [operator=](#) (std::nullptr\_t) noexcept

### 5.429.1 Detailed Description

```
template<typename _Alloc>
struct std::__allocated_ptr<_Alloc>
```

Non-standard RAII type for managing pointers obtained from allocators.

Definition at line 46 of file [allocated\\_ptr.h](#).

### 5.429.2 Constructor & Destructor Documentation

#### 5.429.2.1 \_\_allocated\_ptr() [1/3]

```
template<typename _Alloc>
std::__allocated_ptr<_Alloc>::__allocated_ptr (
    _Alloc & __a,
    pointer __ptr ) [inline], [noexcept]
```

Take ownership of \_\_ptr.

Definition at line 52 of file [allocated\\_ptr.h](#).

## 5.429.2.2 \_\_allocated\_ptr() [2/3]

```
template<typename _Alloc>
template<typename _Ptr, typename _Req = _Require<is_same<_Ptr, value_type*>>>
std::__allocated_ptr<_Alloc>::__allocated_ptr (
    _Alloc & __a,
    _Ptr __ptr ) [inline]
```

Convert \_\_ptr to allocator's pointer type and take ownership of it.

Definition at line 59 of file allocated\_ptr.h.

## 5.429.2.3 \_\_allocated\_ptr() [3/3]

```
template<typename _Alloc>
std::__allocated_ptr<_Alloc>::__allocated_ptr (
    __allocated_ptr<_Alloc> && __gd ) [inline], [noexcept]
```

Transfer ownership of the owned pointer.

Definition at line 65 of file allocated\_ptr.h.

## 5.429.2.4 ~\_\_allocated\_ptr()

```
template<typename _Alloc>
std::__allocated_ptr<_Alloc>::~~__allocated_ptr ( ) [inline]
```

Deallocate the owned pointer.

Definition at line 70 of file allocated\_ptr.h.

References std::allocator\_traits<\_Alloc>::deallocate().

## 5.429.3 Member Function Documentation

## 5.429.3.1 get()

```
template<typename _Alloc>
value_type* std::__allocated_ptr<_Alloc>::get (
    void ) [inline]
```

Get the address that the owned pointer refers to.

Definition at line 85 of file allocated\_ptr.h.

### 5.429.3.2 operator=()

```
template<typename _Alloc >
__allocated_ptr& std::__allocated_ptr< _Alloc >::operator= (
    std::nullptr_t ) [inline], [noexcept]
```

Release ownership of the owned pointer.

Definition at line 78 of file `allocated_ptr.h`.

The documentation for this struct was generated from the following file:

- [allocated\\_ptr.h](#)

## 5.430 std::\_\_atomic\_base< \_ITp > Struct Template Reference

### Public Member Functions

- `__atomic_base` (const `__atomic_base` &)=delete
- constexpr `__atomic_base` (`__int_type` \_\_i) noexcept
- `__attribute__((always_inline))` void store(`__int_type` \_\_i
- bool `is_lock_free` () const noexcept
- bool `is_lock_free` () const volatile noexcept
- `operator __int_type` () const noexcept
- `operator __int_type` () const volatile noexcept
- `__int_type operator&=` (`__int_type` \_\_i) noexcept
- `__int_type operator&=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator++` (int) noexcept
- `__int_type operator++` (int) volatile noexcept
- `__int_type operator++` () noexcept
- `__int_type operator++` () volatile noexcept
- `__int_type operator+=` (`__int_type` \_\_i) noexcept
- `__int_type operator+=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator--` (int) noexcept
- `__int_type operator--` (int) volatile noexcept
- `__int_type operator--` () noexcept
- `__int_type operator--` () volatile noexcept
- `__int_type operator-=` (`__int_type` \_\_i) noexcept
- `__int_type operator-=` (`__int_type` \_\_i) volatile noexcept
- `__atomic_base` & `operator=` (const `__atomic_base` &)=delete
- `__atomic_base` & `operator=` (const `__atomic_base` &) volatile=delete
- `__int_type operator=` (`__int_type` \_\_i) noexcept
- `__int_type operator=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator^=` (`__int_type` \_\_i) noexcept
- `__int_type operator^=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator|=` (`__int_type` \_\_i) noexcept
- `__int_type operator|=` (`__int_type` \_\_i) volatile noexcept

## 5.430.1 Detailed Description

```
template<typename _ITp>
struct std::__atomic_base< _ITp >
```

Base class for atomic integrals.

Definition at line 120 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 5.431 std::\_\_atomic\_base&lt; \_PTp \* &gt; Struct Template Reference

## Public Member Functions

- **\_\_atomic\_base** (const [\\_\\_atomic\\_base](#) &)=delete
- constexpr **\_\_atomic\_base** (\_\_pointer\_type \_\_p) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_pointer\_type \_\_p
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_pointer\_type** () const noexcept
- **operator \_\_pointer\_type** () const volatile noexcept
- \_\_pointer\_type **operator++** (int) noexcept
- \_\_pointer\_type **operator++** (int) volatile noexcept
- \_\_pointer\_type **operator++** () noexcept
- \_\_pointer\_type **operator++** () volatile noexcept
- \_\_pointer\_type **operator+=** (ptrdiff\_t \_\_d) noexcept
- \_\_pointer\_type **operator+=** (ptrdiff\_t \_\_d) volatile noexcept
- \_\_pointer\_type **operator--** (int) noexcept
- \_\_pointer\_type **operator--** (int) volatile noexcept
- \_\_pointer\_type **operator--** () noexcept
- \_\_pointer\_type **operator--** () volatile noexcept
- \_\_pointer\_type **operator-=** (ptrdiff\_t \_\_d) noexcept
- \_\_pointer\_type **operator-=** (ptrdiff\_t \_\_d) volatile noexcept
- [\\_\\_atomic\\_base](#) & **operator=** (const [\\_\\_atomic\\_base](#) &)=delete
- [\\_\\_atomic\\_base](#) & **operator=** (const [\\_\\_atomic\\_base](#) &) volatile=delete
- \_\_pointer\_type **operator=** (\_\_pointer\_type \_\_p) noexcept
- \_\_pointer\_type **operator=** (\_\_pointer\_type \_\_p) volatile noexcept

## 5.431.1 Detailed Description

```
template<typename _PTp>
struct std::__atomic_base< _PTp * >
```

Partial specialization for pointer types.

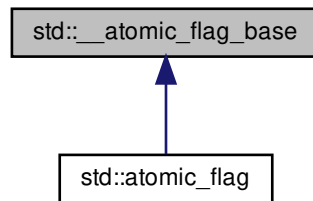
Definition at line 565 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

### 5.432 `std::__atomic_flag_base` Struct Reference

Inheritance diagram for `std::__atomic_flag_base`:



#### Public Attributes

- `__atomic_flag_data_type _M_i`

#### 5.432.1 Detailed Description

Base type for `atomic_flag`.

Base type is POD with data, allowing `atomic_flag` to derive from it and meet the standard layout type requirement. In addition to compatibility with a C interface, this allows different implementations of `atomic_flag` to use the same atomic operation functions, via a standard conversion to the `__atomic_flag_base` argument.

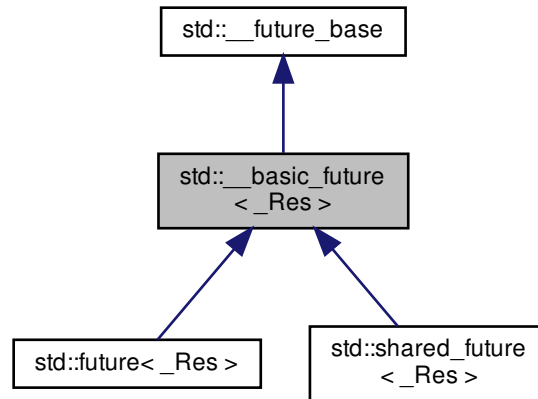
Definition at line 150 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 5.433 std::\_\_basic\_future&lt; \_Res &gt; Class Template Reference

Inheritance diagram for std::\_\_basic\_future< \_Res >:



## Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr< _Res, \_Result\_base::\_Deleter >`
- `using \_State\_base = \_State\_baseV2`

## Public Member Functions

- `\_\_basic\_future (const \_\_basic\_future &)=delete`
- `\_\_basic\_future & operator= (const \_\_basic\_future &)=delete`
- `bool valid () const noexcept`
- `void wait () const`
- `template<typename _Rep, typename _Period >`  
`future\_status wait\_for (const chrono::duration< _Rep, _Period > &__rel) const`
- `template<typename _Clock, typename _Duration >`  
`future\_status wait\_until (const chrono::time\_point< _Clock, _Duration > &__abs) const`

## Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static \_Ptr< \_Result\_alloc< _Res, _Allocator > > \_S\_allocate\_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static \_Ptr< \_Result< _Res > > \_S\_allocate\_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared\_ptr< \_State\_base > \_S\_make\_async\_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared\_ptr< \_State\_base > \_S\_make\_deferred\_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static \_Task\_setter< _Res_ptr, _BoundFn > \_S\_task\_setter (_Res_ptr &__ptr, _BoundFn &__call)`



### Protected Types

- typedef `__future_base::Result`< \_Res > & `__result_type`
- typedef `shared_ptr`< \_State\_base > `__state_type`

### Protected Member Functions

- `__basic_future` (const `__state_type` &\_\_state)
- `__basic_future` (const `shared_future`< \_Res > &) noexcept
- `__basic_future` (`shared_future`< \_Res > &&) noexcept
- `__basic_future` (`future`< \_Res > &&) noexcept
- `__result_type _M_get_result` () const
- void `_M_swap` (`__basic_future` &\_\_that) noexcept

#### 5.433.1 Detailed Description

```
template<typename _Res>
class std::__basic_future< _Res >
```

Common implementation for future and shared\_future.

Definition at line 671 of file future.

#### 5.433.2 Member Typedef Documentation

##### 5.433.2.1 \_Ptr

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file future.

#### 5.433.3 Member Function Documentation

## 5.433.3.1 \_M\_get\_result()

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result ( ) const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

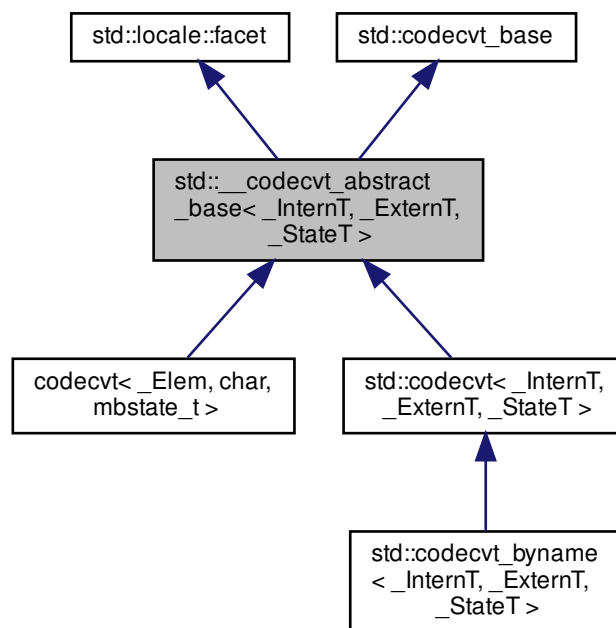
Definition at line 714 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.434 std::\_\_codecvt\_abstract\_base&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Protected Member Functions

- **\_\_codecvt\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const =0 throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 5.434.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >
```

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 68 of file codecvt.h.

## 5.434.2 Member Function Documentation

5.434.2.1 `do_out()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >::do\_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [protected], [pure virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`out` for more information.

Implemented in [std::codecvt< char32\\_t, char, mbstate\\_t >](#), [std::codecvt< char16\\_t, char, mbstate\\_t >](#), [std::codecvt< wchar\\_t, char, mbstate\\_t >](#), [std::codecvt< char, char, mbstate\\_t >](#), [std::codecvt< \\_InternT, \\_ExternT, \\_StateT >](#), [std::codecvt< \\_Elem, char, mbstate\\_t >](#), and [std::codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#).

Referenced by [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >::out\(\)](#).

5.434.2.2 `in()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

**5.434.2.3 out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

5.434.2.4 `unshift()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

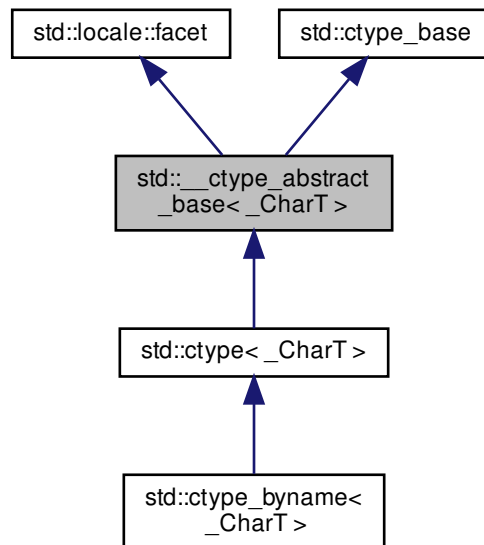
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**5.435 std::\_\_ctype\_abstract\_base<\_CharT> Class Template Reference**

Inheritance diagram for std::\_\_ctype\_abstract\_base<\_CharT>:

**Public Types**

- typedef const int \* **\_\_to\_type**
- typedef \_CharT [char\\_type](#)
- typedef unsigned short **mask**

## Public Member Functions

- bool [is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_default) const
- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_default, char \* \_\_to) const
- const [char\\_type](#) \* [scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- [\\_\\_ctype\\_abstract\\_base](#) (size\_t \_\_refs=0)
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const =0
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const =0
- virtual char [do\\_narrow](#) ([char\\_type](#) \_\_c, char \_\_default) const =0
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_default, char \* \_\_to) const =0
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const =0
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const =0
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const =0
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const =0
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const =0



### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### 5.435.1 Detailed Description

```
template<typename _CharT>
class std::__ctype_abstract_base< _CharT >
```

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 150 of file locale\_facets.h.

#### 5.435.2 Member Typedef Documentation

##### 5.435.2.1 char\_type

```
template<typename _CharT>
typedef _CharT std::__ctype_abstract_base< _CharT >::char_type
```

Typedef for the template parameter.

Definition at line 155 of file locale\_facets.h.

#### 5.435.3 Member Function Documentation

##### 5.435.3.1 do\_is() [1/2]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
    mask __m,
    char_type __c ) const [protected], [pure virtual]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

<a href="#"><code>__c</code></a>	The char_type to find the mask of.
<a href="#"><code>__m</code></a>	The mask to compare against.

## Returns

(M & \_\_m) != 0.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::is\(\)](#).

## 5.435.3.2 do\_is() [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base<_CharT>::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [protected], [pure virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

<a href="#"><code>__lo</code></a>	Pointer to start of range.
<a href="#"><code>__hi</code></a>	Pointer to end of range.
<a href="#"><code>__vec</code></a>	Pointer to an array of mask storage.

## Returns

[`\_\_hi`](#).

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

### 5.435.3.3 `do_narrow()` [1/2]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
    char_type __c,
    char __default ) const [protected], [pure virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__default</code>	Char to return if conversion fails.

#### Returns

The converted `char`.

Implemented in `std::ctype< wchar_t >`, and `std::ctype< _CharT >`.

Referenced by `std::__ctype_abstract_base< wchar_t >::narrow()`.

### 5.435.3.4 `do_narrow()` [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base< _CharT >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to ) const [protected], [pure virtual]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

## 5.435.3.5 do\_scan\_is()

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base<_CharT>::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [pure virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to a matching char\_type if found, else `__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_is()`.

### 5.435.3.6 do\_scan\_not()

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base< _CharT >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [pure virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to a non-matching char\_type if found, else `__hi`.

Implemented in `std::ctype< wchar_t >`, and `std::ctype< _CharT >`.

Referenced by `std::__ctype_abstract_base< wchar_t >::scan_not()`.

### 5.435.3.7 do\_tolower() [1/2]


```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
    char_type __c ) const [protected], [pure virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

<a href="#"></a> <code>__c</code>	The char_type to convert.
--	---------------------------

## Returns

The lowercase char\_type if convertible, else `__c`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::tolower\(\)](#).

## 5.435.3.8 do\_tolower() [2/2]



```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base<_CharT>::do_tolower (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [pure virtual]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

<a href="#"></a> <code>__lo</code>	Pointer to start of range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

## 5.435.3.9 do\_toupper() [1/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base<_CharT>::do_toupper (
    char_type __c ) const    [protected], [pure virtual]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

#### Returns

The uppercase `char_type` if convertible, else `__c`.

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

Referenced by `std::__ctype_abstract_base< wchar_t >::toupper()`.

#### 5.435.3.10 do\_toupper() [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base< _CharT >::do_toupper (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [pure virtual]
```

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

## 5.435.3.11 do\_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_widen (
    char __c ) const [protected], [pure virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

## Returns

The converted char\_type

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >::widen\(\)](#).

## 5.435.3.12 do\_widen() [2/2]

```
template<typename _CharT>
virtual const char* std::__ctype_abstract_base< _CharT >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to ) const [protected], [pure virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.



**Parameters**

<a href="#"><code>↵</code></a> <code>_lo</code>	Pointer to start range.
<a href="#"><code>↵</code></a> <code>_hi</code>	Pointer to end of range.
<a href="#"><code>↵</code></a> <code>_to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

**5.435.3.13 is() [1/2]**

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
    mask __m,
    char_type __c ) const [inline]
```

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵type>::do_is()`.

**Parameters**

<a href="#"><code>↵</code></a> <code>_c</code>	The char_type to compare the mask of.
<a href="#"><code>↵</code></a> <code>_m</code>	The mask to compare against.

**Returns**

`(M & __m) != 0`.

Definition at line 169 of file locale\_facets.h.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

## 5.435.3.14 is() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

## Returns

`__hi`.

Definition at line 186 of file locale\_facets.h.

## 5.435.3.15 narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT>::narrow (
    char_type __c,
    char __default ) const [inline]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The char_type to convert.
<code>__default</code>	Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 331 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter >::get()`, and `std::time_put<_CharT, _OutIter >::put()`.

**5.435.3.16 narrow()** [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to ) const [inline]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 353 of file locale\_facets.h.




**5.435.3.17 scan\_is()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT >::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [inline]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

## Parameters

<a href="#"></a> <code>__m</code>	The mask to compare against.
<a href="#"></a> <code>__lo</code>	Pointer to start of range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.

## Returns

Pointer to matching char\_type if found, else `__hi`.

Definition at line 202 of file locale\_facets.h.




## 5.435.3.18 scan\_not()

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

## Parameters

<a href="#"></a> <code>__m</code>	The mask to compare against.
<a href="#"></a> <code>__lo</code>	Pointer to first char in range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.

## Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 218 of file locale\_facets.h.

**5.435.3.19** `tolower()` [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
    char_type __c ) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

<code>__c</code>	The char_type to convert.
------------------	---------------------------

**Returns**

The lowercase char\_type if convertible, else `__c`.

Definition at line 261 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

**5.435.3.20** `tolower()` [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::tolower (
    char_type * __lo,
    const char_type * __hi ) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 276 of file locale\_facets.h.


#### 5.435.3.21 toupper() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
    char_type __c ) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

##### Parameters

 __c	The char_type to convert.
--	---------------------------

##### Returns

The uppercase char\_type if convertible, else \_\_c.

Definition at line 232 of file locale\_facets.h.

Referenced by std::time\_get< \_CharT, \_InIter >::get().



#### 5.435.3.22 toupper() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::toupper (
    char_type * __lo,
    const char_type * __hi ) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

##### Parameters

 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 247 of file `locale_facets.h`.

**5.435.3.23 `widen()`** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
    char __c ) const [inline]
```

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The <code>char</code> to convert.
------------------	-----------------------------------

**Returns**

The converted `char_type`.

Definition at line 293 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::money_get< _CharT, _InIter >::do_get()`, `std::time_put< _CharT, _OutIter >::do_put()`, and `std::money_put< _CharT, _OutIter >::do_put()`.

**5.435.3.24 `widen()`** [2/2]




```
template<typename _CharT>
const char* std::__ctype_abstract_base< _CharT >::widen (
    const char * __lo,
    const char * __hi,
    char_type * __to ) const [inline]
```

Widen array to `char_type`.

This function converts each `char` in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<a href="#"></a> <code>__lo</code>	Pointer to start of range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.
<a href="#"></a> <code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 312 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.436 std::\_\_debug::bitset&lt; \_Nb &gt; Class Template Reference

Inherits `bitset< _Nb >`.

## Public Types

- typedef `_Base::reference` **reference**

## Public Member Functions

- constexpr **bitset** (unsigned long long \_\_val) noexcept
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, [!\[\]\(f0543fe51acd79be3858008749d93a88\_img.jpg\)](#) Traits, \_Alloc >::size\_type \_\_pos=0, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type [!\[\]\(b452a1210835992e25e075124622531b\_img.jpg\)](#) n=([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::npos))
- template<class \_CharT, class \_Traits, class \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT [!\[\]\(7bc2b99ff222bd0a25e1cf77d692b0e7\_img.jpg\)](#) zero, \_CharT [!\[\]\(2a8e852b9c8306ea71a116afe600d421\_img.jpg\)](#) one=\_CharT('1'))
- **bitset** (const [\\_Base](#) &\_\_x)
- template<typename \_CharT >  
**bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT [!\[\]\(ae4b925a6b10795b21150db3177cb91b\_img.jpg\)](#) zero=\_CharT('0'), \_CharT [!\[\]\(5797a37c829d4ce48ff4c5017f3d10fa\_img.jpg\)](#) one=\_CharT('1'))
- [\\_Base](#) & **M\_base** () noexcept
- const [\\_Base](#) & **M\_base** () const noexcept
- [bitset](#)< \_Nb > & **flip** () noexcept
- [bitset](#)< \_Nb > & **flip** (size\_t \_\_pos)
- bool **operator!=** (const [bitset](#)< \_Nb > &\_\_rhs) const noexcept



- `bitset<_Nb> & operator&=` (const `bitset<_Nb> &__rhs`) noexcept
- `bitset<_Nb> operator<<` (size\_t \_\_pos) const noexcept
- `bitset<_Nb> & operator<<=` (size\_t \_\_pos) noexcept
- `bool operator==` (const `bitset<_Nb> &__rhs`) const noexcept
- `bitset<_Nb> operator>>` (size\_t \_\_pos) const noexcept
- `bitset<_Nb> & operator>>=` (size\_t \_\_pos) noexcept
- reference `operator[]` (size\_t \_\_pos)
- constexpr `bool operator[]` (size\_t \_\_pos) const
- `bitset<_Nb> & operator^=` (const `bitset<_Nb> &__rhs`) noexcept
- `bitset<_Nb> & operator|=` (const `bitset<_Nb> &__rhs`) noexcept
- `bitset<_Nb> operator~` () const noexcept
- `bitset<_Nb> & reset` () noexcept
- `bitset<_Nb> & reset` (size\_t \_\_pos)
- `bitset<_Nb> & set` () noexcept
- `bitset<_Nb> & set` (size\_t \_\_pos, bool \_\_val=true)
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`std::basic_string<_CharT, _Traits, _Alloc> to_string` () const
- `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_string<_CharT, _Traits, _Alloc> to_string` (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- `template<typename _CharT, typename _Traits>`  
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string` () const
- `template<class _CharT, class _Traits>`  
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string` (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- `template<typename _CharT>`  
`std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string` () const
- `template<class _CharT>`  
`std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string` (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string` () const
- `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string` (char \_\_zero, char \_\_one='1') const

#### 5.436.1 Detailed Description

```
template<size_t _Nb>
class std::__debug::bitset<_Nb>
```

Class `std::bitset` with additional safety/checking/debug instrumentation.

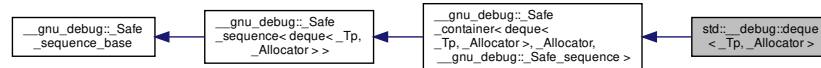
Definition at line 44 of file `debug/bitset`.

The documentation for this class was generated from the following file:

- `debug/bitset`

## 5.437 std::\_\_debug::deque&lt; \_Tp, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::deque< \_Tp, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, deque >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, deque >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **deque** (const `deque` &)=default
- **deque** (`deque` &&)=default
- **deque** (const `deque` &\_\_d, const `_Allocator` &\_\_a)
- **deque** (`deque` && \_\_d, const `_Allocator` &\_\_a)
- **deque** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a=allocator\_type())
- **deque** (const `_Allocator` &\_\_a)
- **deque** (size\_type \_\_n, const `_Allocator` &\_\_a=\_Allocator())
- **deque** (size\_type \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=\_Allocator())
- template<class `_InputIterator` , typename = std::RequireInputIter<\_InputIterator>>>  
**deque** (\_InputIterator \_\_first, \_InputIterator \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- **deque** (const `_Base` &\_\_x)
- `_Base` & **M\_base** () noexcept
- const `_Base` & **M\_base** () const noexcept
- void **M\_invalidate\_if** (\_Predicate \_\_pred)
- void **M\_swap** (\_Safe\_container &\_\_x) noexcept
- void **M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- template<class `_InputIterator` , typename = std::RequireInputIter<\_InputIterator>>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_t)
- void **assign** (`initializer_list`< `value_type` > \_\_l)
- reference **back** () noexcept

- `const_reference back ()` `const noexcept`
- `iterator begin ()` `noexcept`
- `const_iterator begin ()` `const noexcept`
- `const_iterator cbegin ()` `const noexcept`
- `const_iterator cend ()` `const noexcept`
- `void clear ()` `noexcept`
- `const_reverse_iterator crbegin ()` `const noexcept`
- `const_reverse_iterator crend ()` `const noexcept`
- `template<typename... _Args>`  
`iterator emplace (const_iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_back (_Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_front (_Args &&... __args)`
- `iterator end ()` `noexcept`
- `const_iterator end ()` `const noexcept`
- `iterator erase (const_iterator __position)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `reference front ()` `noexcept`
- `const_reference front ()` `const noexcept`
- `iterator insert (const_iterator __position, const _Tp &__x)`
- `iterator insert (const_iterator __position, _Tp &&__x)`
- `iterator insert (const_iterator __position, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const _Tp &__x)`
- `template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `deque & operator= (const deque &)=default`
- `deque & operator= (deque &&)=default`
- `deque & operator= (initializer_list< value_type > __l)`
- `reference operator[] (size_type __n)` `noexcept`
- `const_reference operator[] (size_type __n)` `const noexcept`
- `void pop_back ()` `noexcept`
- `void pop_front ()` `noexcept`
- `void push_back (const _Tp &__x)`
- `void push_back (_Tp &&__x)`
- `void push_front (const _Tp &__x)`
- `void push_front (_Tp &&__x)`
- `reverse_iterator rbegin ()` `noexcept`
- `const_reverse_iterator rbegin ()` `const noexcept`
- `reverse_iterator rend ()` `noexcept`
- `const_reverse_iterator rend ()` `const noexcept`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const _Tp &__c)`
- `void shrink_to_fit ()` `noexcept`
- `void swap (deque &__x)` `noexcept` `(/*conditional */)`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x) noexcept

## 5.437.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::deque<_Tp, _Allocator>
```

Class std::deque with safety/checking/debug instrumentation.

Definition at line 45 of file debug/deque.

## 5.437.2 Member Function Documentation

5.437.2.1 [\\_M\\_detach\\_all\(\)](#)

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::~~\\_Safe\\_sequence\\_base\(\)](#).

5.437.2.2 [\\_M\\_detach\\_singular\(\)](#)

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, `i->_M_version == _M_version`.

### 5.437.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( )    [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.437.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const    [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

### 5.437.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::_M_invalidate_if (   
    _Predicate __pred )    [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

### 5.437.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )    [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

### 5.437.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (   
    _Safe_sequence_base & __x )    [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.437.2.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::_M_transfer_from_if (
    _Safe_sequence< deque< _Tp, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

## 5.437.3 Member Data Documentation

## 5.437.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.437.3.2 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.437.3.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

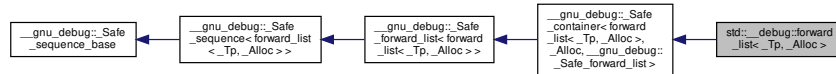
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/deque](#)

## 5.438 std::\_\_debug::forward\_list< \_Tp, \_Alloc > Class Template Reference

Inheritance diagram for std::\_\_debug::forward\_list< \_Tp, \_Alloc >:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug__::Safe_iterator< _Base_const_iterator, forward_list >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug__::Safe_iterator< _Base_iterator, forward_list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- **forward\_list** (const allocator\_type &\_\_al) noexcept
- **forward\_list** (const forward\_list &\_\_list, const allocator\_type &\_\_al)
- **forward\_list** (forward\_list &&\_\_list, const allocator\_type &\_\_al)
- **forward\_list** (size\_type \_\_n, const allocator\_type &\_\_al=allocator\_type())
- **forward\_list** (size\_type \_\_n, const \_Tp &\_\_value, const allocator\_type &\_\_al=allocator\_type())
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
**forward\_list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_al=allocator\_type())
- **forward\_list** (const forward\_list &)=default
- **forward\_list** (forward\_list &&)=default
- **forward\_list** (std::initializer\_list< \_Tp > \_\_il, const allocator\_type &\_\_al=allocator\_type())
- `_Base & _M_base` () noexcept
- const `_Base & _M_base` () const noexcept
- void `_M_invalidate_if` (\_Predicate \_\_pred)
- void `_M_swap` (\_Safe\_container &\_\_x) noexcept
- void `_M_transfer_from_if` (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_val)
- void **assign** (std::initializer\_list< \_Tp > \_\_il)
- **iterator before\_begin** () noexcept
- **const\_iterator before\_begin** () const noexcept
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept

- [const\\_iterator cbefore\\_begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- template<typename... \_Args>  
[iterator emplace\\_after](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [iterator erase\\_after](#) (const\_iterator \_\_pos)
- [iterator erase\\_after](#) (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- [iterator insert\\_after](#) (const\_iterator \_\_pos, const \_Tp &\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, \_Tp &&\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- template<typename \_InputIterator, typename = std::\_\_RequireInputIter<\_InputIterator>>  
[iterator insert\\_after](#) (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, std::initializer\_list< \_Tp > \_\_il)
- void [merge](#) (forward\_list &&\_\_list)
- void [merge](#) (forward\_list &\_\_list)
- template<typename \_Comp >  
void [merge](#) (forward\_list &&\_\_list, \_Comp \_\_comp)
- template<typename \_Comp >  
void [merge](#) (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & [operator=](#) (const forward\_list &)=default
- forward\_list & [operator=](#) (forward\_list &&)=default
- forward\_list & [operator=](#) (std::initializer\_list< \_Tp > \_\_il)
- void [pop\\_front](#) ()
- void [remove](#) (const \_Tp &\_\_val)
- template<typename \_Pred >  
void [remove\\_if](#) (\_Pred \_\_pred)
- void [resize](#) (size\_type \_\_sz)
- void [resize](#) (size\_type \_\_sz, const value\_type &\_\_val)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&\_\_list)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &\_\_list)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void [splice\\_after](#) (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void [swap](#) (forward\_list &\_\_list) noexcept(noexcept(declval< \_Base & >().swap(\_\_list)))
- void [unique](#) ()
- template<typename \_BinPred >  
void [unique](#) (\_BinPred \_\_binary\_pred)

#### Public Attributes

- \_Safe\_iterator\_base \* [\\_M\\_const\\_iterators](#)
- \_Safe\_iterator\_base \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)



### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- void `_M_invalidate_all` ()
- void `_M_invalidate_all` () const
- void `_M_revalidate_singular` ()
- `_Safe_container & _M_safe` () noexcept
- void `_M_swap` (\_Safe\_sequence\_base &) noexcept

#### 5.438.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__debug::forward_list<_Tp, _Alloc >
```

Class `std::forward_list` with safety/checking/debug instrumentation.

Definition at line 184 of file `debug/forward_list`.

#### 5.438.2 Member Function Documentation

##### 5.438.2.1 `_M_detach_all`()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base`().

##### 5.438.2.2 `_M_detach_singular`()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

## 5.438.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( )    [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.438.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const    [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

## 5.438.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< forward_list< _Tp, _Alloc > >::_M_invalidate_if (
    _Predicate __pred )    [inherited]
```

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

## 5.438.2.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )    [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.438.2.7 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< forward_list< _Tp, _Alloc > >::_M_transfer_from_if (
    _Safe_sequence< forward_list< _Tp, _Alloc > > & __from,
    _Predicate __pred )    [inherited]
```

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe\_sequence.tcc.

### 5.438.3 Member Data Documentation

#### 5.438.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.438.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.438.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

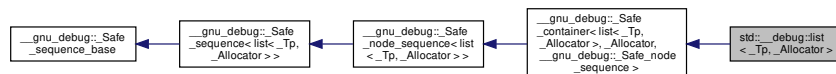
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

### 5.439 `std::__debug::list< _Tp, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::list< _Tp, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, list >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **list** (const `list` &)=default
- **list** (`list` &&)=default
- **list** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a=`allocator_type`())
- **list** (const `list` &\_\_x, const `allocator_type` &\_\_a)
- **list** (`list` &&\_\_x, const `allocator_type` &\_\_a)
- **list** (const `_Allocator` &\_\_a) noexcept
- **list** (`size_type` \_\_n, const `allocator_type` &\_\_a=`allocator_type`())
- **list** (`size_type` \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=`_Allocator`())
- template<class `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`>  
**list** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=`_Allocator`())
- **list** (const `_Base` &\_\_x)
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_swap** (`_Safe_container` &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- void **assign** (`initializer_list`< `value_type` > \_\_l)
- template<class `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`>  
void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (`size_type` \_\_n, const `_Tp` &\_\_t)
- reference **back** () noexcept
- const\_reference **back** () const noexcept
- **iterator begin** () noexcept
- `const_iterator` **begin** () const noexcept
- `const_iterator` **cbegin** () const noexcept
- `const_iterator` **cend** () const noexcept
- void **clear** () noexcept
- `const_reverse_iterator` **crbegin** () const noexcept
- `const_reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`>  
`iterator` **emplace** (`const_iterator` \_\_position, `_Args` &&... \_\_args)
- **iterator end** () noexcept
- `const_iterator` **end** () const noexcept

- `iterator erase (const_iterator __position)` noexcept
- `iterator erase (const_iterator __first, const_iterator __last)` noexcept
- reference `front ()` noexcept
- const\_reference `front ()` const noexcept
- `iterator insert (const_iterator __position, const_Tp &__x)`
- `iterator insert (const_iterator __position, Tp &&__x)`
- `iterator insert (const_iterator __p, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const_Tp &__x)`
- `template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- void `merge (list &&__x)`
- void `merge (list &__x)`
- `template<class _Compare >`  
void `merge (list &&__x, _Compare __comp)`
- `template<typename _Compare >`  
void `merge (list &__x, _Compare __comp)`
- `list & operator= (const list &)=default`
- `list & operator= (list &&)=default`
- `list & operator= (initializer_list< value_type > __l)`
- void `pop_back ()` noexcept
- void `pop_front ()` noexcept
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rbegin ()` const noexcept
- void `remove (const_Tp &__value)`
- `template<class _Predicate >`  
void `remove_if (_Predicate __pred)`
- `reverse_iterator rend ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- void `resize (size_type __sz)`
- void `resize (size_type __sz, const_Tp &__c)`
- void `sort ()`
- `template<typename _StrictWeakOrdering >`  
void `sort (_StrictWeakOrdering __pred)`
- void `splice (const_iterator __position, list &&__x)` noexcept
- void `splice (const_iterator __position, list &__x)` noexcept
- void `splice (const_iterator __position, list &&__x, const_iterator __i)` noexcept
- void `splice (const_iterator __position, list &__x, const_iterator __i)` noexcept
- void `splice (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last)` noexcept
- void `splice (const_iterator __position, list &__x, const_iterator __first, const_iterator __last)` noexcept
- void `swap (list &__x)` noexcept(*/\*conditional \*/*)
- void `unique ()`
- `template<class _BinaryPredicate >`  
void `unique (_BinaryPredicate __binary_pred)`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- unsigned int `_M_version`

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) & \_\_x) noexcept

## 5.439.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::list< _Tp, _Allocator >
```

Class std::list with safety/checking/debug instrumentation.

Definition at line 45 of file debug/list.

## 5.439.2 Member Function Documentation

5.439.2.1 [\\_M\\_detach\\_all\(\)](#)

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::~~\\_Safe\\_sequence\\_base\(\)](#).

5.439.2.2 [\\_M\\_detach\\_singular\(\)](#)

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators i attached to this sequence, `i->_M_version == _M_version`.

#### 5.439.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( )    [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.439.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const    [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 5.439.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< list< _Tp, _Allocator > >::_M_invalidate_if (   
    _Predicate __pred )    [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 5.439.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )    [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.439.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (   
    _Safe_sequence_base & __x )    [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.439.2.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< list< _Tp, _Allocator > >::_M_transfer_from_if (
    _Safe_sequence< list< _Tp, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

## 5.439.3 Member Data Documentation

## 5.439.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.439.3.2 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.439.3.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

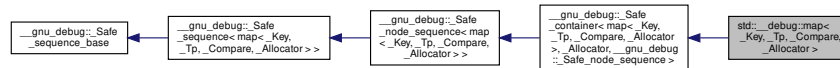
The documentation for this class was generated from the following file:

- [debug/list](#)



## 5.440 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference

Inheritance diagram for std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, map >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, map >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const_Key, _Tp >` **value\_type**

### Public Member Functions

- **map** (const `map` &)=default
- **map** (`map` &&)=default
- **map** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_c=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- **map** (const `allocator_type` &\_\_a)
- **map** (const `map` &\_\_m, const `allocator_type` &\_\_a)
- **map** (`map` &&\_\_m, const `allocator_type` &\_\_a) noexcept(noexcept(`_Base`(std::move(\_\_m.\_M\_base()), \_\_a)))
- **map** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- template<typename `_InputIterator` >  
**map** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a)
- **map** (const `_Base` &\_\_x)
- **map** (const `_Compare` &\_\_comp, const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**map** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)

- void **\_M\_swap** (\_Safe\_container &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- **const\_reverse\_iterator crbegin** () const noexcept
- **const\_reverse\_iterator crend** () const noexcept
- template<typename... \_Args>  
**std::pair**< **iterator**, bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
**iterator emplace\_hint** (**const\_iterator** \_\_pos, \_Args &&... \_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **std::pair**< **iterator**, **iterator** > **equal\_range** (const key\_type &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
**std::pair**< **iterator**, **iterator** > **equal\_range** (const \_Kt &\_\_x)
- **std::pair**< **const\_iterator**, **const\_iterator** > **equal\_range** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
**std::pair**< **const\_iterator**, **const\_iterator** > **equal\_range** (const \_Kt &\_\_x) const
- **iterator erase** (**const\_iterator** \_\_position)
- **iterator erase** (**iterator** \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- **iterator erase** (**const\_iterator** \_\_first, **const\_iterator** \_\_last)
- **iterator find** (const key\_type &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
**iterator find** (const \_Kt &\_\_x)
- **const\_iterator find** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
**const\_iterator find** (const \_Kt &\_\_x) const
- **std::pair**< **iterator**, bool > **insert** (const value\_type &\_\_x)
- **std::pair**< **iterator**, bool > **insert** (value\_type &&\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
**std::pair**< **iterator**, bool > **insert** (\_Pair &&\_\_x)
- void **insert** (std::initializer\_list< value\_type > \_\_list)
- **iterator insert** (**const\_iterator** \_\_position, const value\_type &\_\_x)
- **iterator insert** (**const\_iterator** \_\_position, value\_type &&\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
**iterator insert** (**const\_iterator** \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **iterator lower\_bound** (const key\_type &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
**iterator lower\_bound** (const \_Kt &\_\_x)
- **const\_iterator lower\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
**const\_iterator lower\_bound** (const \_Kt &\_\_x) const
- **map** & **operator=** (const **map** &)=default
- **map** & **operator=** (**map** &&)=default
- **map** & **operator=** (initializer\_list< value\_type > \_\_l)

- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **swap** ([map](#) &\_\_x) noexcept(*/\*conditional \*/*)
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator](#) **upper\_bound** (const \_Kt &\_\_x)
- [const\\_iterator](#) **upper\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator](#) **upper\_bound** (const \_Kt &\_\_x) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x) noexcept

#### 5.440.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::map<_Key, _Tp, _Compare, _Allocator>
```

Class std::map with safety/checking/debug instrumentation.

Definition at line 44 of file debug/map.h.

#### 5.440.2 Member Function Documentation

#### 5.440.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

#### 5.440.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

#### 5.440.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 5.440.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

#### 5.440.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (
    _Predicate __pred ) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 5.440.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.440.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 5.440.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if
(
    _Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 5.440.3 Member Data Documentation

#### 5.440.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.440.3.2 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.440.3.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file safe\_base.h.

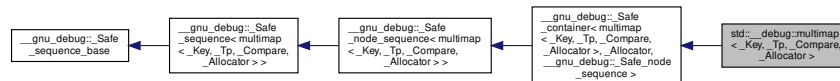
Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_invalidate\_all().

The documentation for this class was generated from the following file:

- [debug/map.h](#)

## 5.441 std::\_\_debug::multimap&lt; \_Key, \_Tp, \_Compare, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >:



## Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_const\_iterator, multimap > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef std::reverse\_iterator< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_iterator, multimap > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef std::reverse\_iterator< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef std::pair< const \_Key, \_Tp > **value\_type**

## Public Member Functions

- **multimap** (const [multimap](#) &)=default
- **multimap** ([multimap](#) &&)=default
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [\\_Compare](#) &\_\_c=\_Compare(), const [allocator\\_type](#) &\_\_a=allocator\_type())
- **multimap** (const [allocator\\_type](#) &\_\_a)
- **multimap** (const [multimap](#) &\_\_m, const [allocator\\_type](#) &\_\_a)
- **multimap** ([multimap](#) &&\_\_m, const [allocator\\_type](#) &\_\_a) noexcept(noexcept([\\_Base](#)(std::move(\_\_m.\_M\_base()), \_\_a)))
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- template<typename [\\_InputIterator](#) >  
**multimap** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [allocator\\_type](#) &\_\_a)
- **multimap** (const [\\_Compare](#) &\_\_comp, const [\\_Allocator](#) &\_\_a=\_Allocator())
- template<typename [\\_InputIterator](#) >  
**multimap** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Compare](#) &\_\_comp=\_Compare(), const [\\_Allocator](#) &\_\_a=\_Allocator())
- **multimap** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_swap](#) ([\\_Safe\\_container](#) &\_\_x) noexcept
- void [\\_M\\_transfer\\_from\\_if](#) ([\\_Safe\\_sequence](#) &\_\_from, [\\_Predicate](#) \_\_pred)
- [iterator](#) **begin** () noexcept
- [const\\_iterator](#) **begin** () const noexcept
- [const\\_iterator](#) **cbegin** () const noexcept
- [const\\_iterator](#) **end** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- template<typename... [\\_Args](#)>  
[iterator](#) **emplace** ([\\_Args](#) &&... \_\_args)
- template<typename... [\\_Args](#)>  
[iterator](#) **emplace\_hint** ([const\\_iterator](#) \_\_pos, [\\_Args](#) &&... \_\_args)
- [iterator](#) **end** () noexcept
- [const\\_iterator](#) **end** () const noexcept
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const [key\\_type](#) &\_\_x)
- template<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const [\\_Kt](#) &\_\_x)
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const [key\\_type](#) &\_\_x) const
- template<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const [\\_Kt](#) &\_\_x) const
- [iterator](#) **erase** ([const\\_iterator](#) \_\_position)
- [iterator](#) **erase** ([iterator](#) \_\_position)
- [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- [iterator](#) **find** (const [key\\_type](#) &\_\_x)
- template<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[iterator](#) **find** (const [\\_Kt](#) &\_\_x)
- [const\\_iterator](#) **find** (const [key\\_type](#) &\_\_x) const
- template<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[const\\_iterator](#) **find** (const [\\_Kt](#) &\_\_x) const

- [iterator insert](#) (const [value\\_type](#) &\_\_x)
- [iterator insert](#) ([value\\_type](#) &&\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
[iterator insert](#) (\_Pair &&\_\_x)
- void [insert](#) (std::initializer\_list< [value\\_type](#) > \_\_list)
- [iterator insert](#) (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- [iterator insert](#) (const [iterator](#) \_\_position, [value\\_type](#) &&\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
[iterator insert](#) (const [iterator](#) \_\_position, \_Pair &&\_\_x)
- template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator lower\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator lower\\_bound](#) (const \_Kt &\_\_x)
- [const\\_iterator lower\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator lower\\_bound](#) (const \_Kt &\_\_x) const
- [multimap](#) & [operator=](#) (const [multimap](#) &)=default
- [multimap](#) & [operator=](#) ([multimap](#) &&)=default
- [multimap](#) & [operator=](#) (initializer\_list< [value\\_type](#) > \_\_l)
- [reverse\\_iterator rbegin](#) () noexcept
- [const\\_reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- void [swap](#) ([multimap](#) &\_\_x) noexcept(*/\*conditional \*/*)
- [iterator upper\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator upper\\_bound](#) (const \_Kt &\_\_x)
- [const\\_iterator upper\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator upper\\_bound](#) (const \_Kt &\_\_x) const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- \_\_gnu\_cxx::\_\_mutex & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x) noexcept



### 5.441.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>>
class std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>
```

Class std::multimap with safety/checking/debug instrumentation.

Definition at line 44 of file debug/multimap.h.

### 5.441.2 Member Function Documentation

#### 5.441.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~Safe\_sequence\_base().

#### 5.441.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

#### 5.441.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.441.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 5.441.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate↵
_if (
    _Predicate __pred ) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

## 5.441.2.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.441.2.7 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.441.2.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_transfer↵
from_if (
    _Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe\_sequence.tcc.

### 5.441.3 Member Data Documentation

#### 5.441.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.441.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.441.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

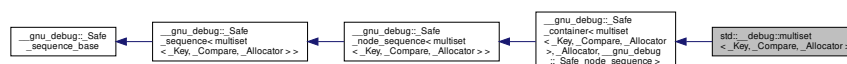
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

### 5.442 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, multiset >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, multiset >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **multiset** (const `multiset` &)=default
- **multiset** (`multiset` &&)=default
- **multiset** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare`(), const `allocator_type` &\_\_a=`allocator_type`())
- **multiset** (const `allocator_type` &\_\_a)
- **multiset** (const `multiset` &\_\_m, const `allocator_type` &\_\_a)
- **multiset** (`multiset` &&\_\_m, const `allocator_type` &\_\_a) noexcept(noexcept(`_Base`(std::move(\_\_m.\_M\_base()), \_\_a)))
- **multiset** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- `template<typename _InputIterator >`  
**multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a)
- **multiset** (const `_Compare` &\_\_comp, const `_Allocator` &\_\_a=`_Allocator`())
- `template<typename _InputIterator >`  
**multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare`(), const `_Allocator` &\_\_a=`_Allocator`())
- **multiset** (const `_Base` &\_\_x)
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_swap** (`_Safe_container` &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- **iterator begin** () noexcept
- `const_iterator` **begin** () const noexcept
- `const_iterator` **cbegin** () const noexcept
- `const_iterator` **cend** () const noexcept
- void **clear** () noexcept
- `const_reverse_iterator` **crbegin** () const noexcept
- `const_reverse_iterator` **crend** () const noexcept
- `template<typename... _Args >`  
**iterator emplace** (`_Args` &&... \_\_args)

- `template<typename... _Args>`  
`iterator` **emplace\_hint** (`const_iterator` \_\_pos, `_Args` &&... \_\_args)
- `iterator` **end** () noexcept
- `const_iterator` **end** () const noexcept
- `std::pair< iterator, iterator >` **equal\_range** (const key\_type &\_\_x)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (const key\_type &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< iterator, iterator >` **equal\_range** (const \_Kt &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< const_iterator, const_iterator >` **equal\_range** (const \_Kt &\_\_x) const
- `iterator` **erase** (`const_iterator` \_\_position)
- `size_type` **erase** (const key\_type &\_\_x)
- `iterator` **erase** (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator` **find** (const key\_type &\_\_x)
- `const_iterator` **find** (const key\_type &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **find** (const \_Kt &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **find** (const \_Kt &\_\_x) const
- `iterator` **insert** (const value\_type &\_\_x)
- `iterator` **insert** (value\_type &&\_\_x)
- `iterator` **insert** (`const_iterator` \_\_position, const value\_type &\_\_x)
- `iterator` **insert** (`const_iterator` \_\_position, value\_type &&\_\_x)
- `template<typename _InputIterator >`  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (`initializer_list`< value\_type > \_\_l)
- `iterator` **lower\_bound** (const key\_type &\_\_x)
- `const_iterator` **lower\_bound** (const key\_type &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **lower\_bound** (const \_Kt &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **lower\_bound** (const \_Kt &\_\_x) const
- `multiset` & **operator=** (const `multiset` &)=default
- `multiset` & **operator=** (`multiset` &&)=default
- `multiset` & **operator=** (`initializer_list`< value\_type > \_\_l)
- `reverse_iterator` **rbegin** () noexcept
- `const_reverse_iterator` **rbegin** () const noexcept
- `reverse_iterator` **rend** () noexcept
- `const_reverse_iterator` **rend** () const noexcept
- void **swap** (`multiset` &\_\_x) noexcept(*/\*conditional \*/*)
- `iterator` **upper\_bound** (const key\_type &\_\_x)
- `const_iterator` **upper\_bound** (const key\_type &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **upper\_bound** (const \_Kt &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **upper\_bound** (const \_Kt &\_\_x) const

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x) noexcept

#### 5.442.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::multiset<_Key, _Compare, _Allocator >
```

Class std::multiset with safety/checking/debug instrumentation.

Definition at line 44 of file debug/multiset.h.

#### 5.442.2 Member Function Documentation

##### 5.442.2.1 [\\_M\\_detach\\_all](#)()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::~~\\_Safe\\_sequence\\_base](#)().

##### 5.442.2.2 [\\_M\\_detach\\_singular](#)()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

### Postcondition

for all iterators *i* attached to this sequence, *i*->[\\_M\\_version](#) == [\\_M\\_version](#).

#### 5.442.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( )    [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.442.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const    [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 5.442.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_invalidate_if (   
    _Predicate __pred )    [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 5.442.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )    [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.442.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (   
    _Safe_sequence_base & __x )    [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 5.442.2.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_transfer_from_if  
(  
    _Safe_sequence< multiset< _Key, _Compare, _Allocator > > & __from,  
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 5.442.3 Member Data Documentation

#### 5.442.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.442.3.2 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.442.3.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

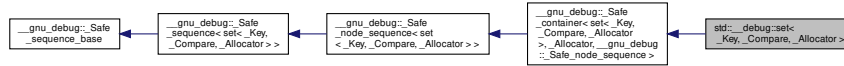
The documentation for this class was generated from the following file:

- [debug/multiset.h](#)



## 5.443 std::\_\_debug::set<\_Key,\_Compare,\_Allocator> Class Template Reference

Inheritance diagram for std::\_\_debug::set<\_Key,\_Compare,\_Allocator>:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_const_iterator, set>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_iterator, set>` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

### Public Member Functions

- **set** (const `set` &)=default
- **set** (`set` &&)=default
- **set** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- **set** (const `allocator_type` &\_\_a)
- **set** (const `set` &\_\_x, const `allocator_type` &\_\_a)
- **set** (`set` &&\_\_x, const `allocator_type` &\_\_a) noexcept(noexcept(`_Base`(std::move(\_\_x.\_M\_base()), \_\_a)))
- **set** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- template<typename `_InputIterator` >  
**set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a)
- **set** (const `_Compare` &\_\_comp, const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **set** (const `_Base` &\_\_x)
- `_Base` & `_M_base` () noexcept
- const `_Base` & `_M_base` () const noexcept
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_swap` (`_Safe_container` &\_\_x) noexcept

- `void _M_transfer_from_if` (`_Safe_sequence &__from, _Predicate __pred`)
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- `template<typename... _Args>`  
`std::pair< iterator, bool > emplace` (`_Args &&... __args`)
- `template<typename... _Args>`  
`iterator emplace_hint` (`const_iterator __pos, _Args &&... __args`)
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `std::pair< iterator, iterator > equal_range` (`const key_type &__x`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< iterator, iterator > equal_range` (`const _Kt &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< const_iterator, const_iterator > equal_range` (`const _Kt &__x`) const
- `iterator erase` (`const_iterator __position`)
- `size_type erase` (`const key_type &__x`)
- `iterator erase` (`const_iterator __first, const_iterator __last`)
- `iterator find` (`const key_type &__x`)
- `const_iterator find` (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator find` (`const _Kt &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator find` (`const _Kt &__x`) const
- `std::pair< iterator, bool > insert` (`const value_type &__x`)
- `std::pair< iterator, bool > insert` (`value_type &&__x`)
- `iterator insert` (`const_iterator __position, const value_type &__x`)
- `iterator insert` (`const_iterator __position, value_type &&__x`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first, _InputIterator __last`)
- `void insert` (`initializer_list< value_type > __l`)
- `iterator lower_bound` (`const key_type &__x`)
- `const_iterator lower_bound` (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator lower_bound` (`const _Kt &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator lower_bound` (`const _Kt &__x`) const
- `set & operator=` (`const set &__x`)=default
- `set & operator=` (`set &&__x`)=default
- `set & operator=` (`initializer_list< value_type > __l`)
- `reverse_iterator rbegin` () noexcept
- `const_reverse_iterator rbegin` () const noexcept
- `reverse_iterator rend` () noexcept
- `const_reverse_iterator rend` () const noexcept
- `void swap` (`set &__x`) noexcept(*/\*conditional \*/*)
- `iterator upper_bound` (`const key_type &__x`)

- [const\\_iterator upper\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator upper\\_bound](#) (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator upper\\_bound](#) (const \_Kt &\_\_x) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x) noexcept

#### 5.443.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::set<_Key, _Compare, _Allocator>
```

Class std::set with safety/checking/debug instrumentation.

Definition at line 44 of file debug/set.h.

#### 5.443.2 Member Function Documentation

##### 5.443.2.1 [\\_M\\_detach\\_all](#)()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base::~~Safe\\_sequence\\_base\(\)](#).

#### 5.443.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

##### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

#### 5.443.2.3 \_M\_get\_mutex()

```
__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 5.443.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

#### 5.443.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_invalidate_if (   
    _Predicate __pred ) [inherited]
```

Invalidates all iterators x that reference this sequence, are not singular, and for which \_\_pred(x) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

#### 5.443.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.443.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 5.443.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_transfer_from_if (
    _Safe_sequence< set< _Key, _Compare, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 5.443.3 Member Data Documentation

#### 5.443.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.443.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.443.3.3 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file safe\_base.h.

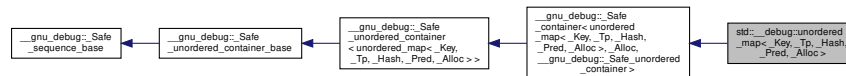
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/set.h](#)

## 5.444 std::\_\_debug::unordered\_map&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inheritance diagram for `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_map >` **const\_iterator**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_map >` **const\_local\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, unordered_map >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_map >` **local\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const [unordered\\_map](#) &)=default
- **unordered\_map** (const [\\_Base](#) &\_\_x)
- **unordered\_map** ([unordered\\_map](#) &&)=default
- **unordered\_map** (const allocator\_type &\_\_a)
- **unordered\_map** (const [unordered\\_map](#) &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** ([unordered\\_map](#) &&\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- [iterator](#) **begin** () noexcept
- const [iterator](#) **begin** () const noexcept
- [local\\_iterator](#) **begin** (size\_type \_\_b)
- const [local\\_iterator](#) **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const [iterator](#) **cbegin** () const noexcept
- const [local\\_iterator](#) **cbegin** (size\_type \_\_b) const
- const [iterator](#) **cend** () const noexcept
- const [local\\_iterator](#) **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_hint, \_Args &&... \_\_args)
- [iterator](#) **end** () noexcept
- const [iterator](#) **end** () const noexcept
- [local\\_iterator](#) **end** (size\_type \_\_b)
- const [local\\_iterator](#) **end** (size\_type \_\_b) const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_key)
- [std::pair](#)< const [iterator](#), const [iterator](#) > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- [iterator](#) **erase** (const [iterator](#) \_\_it)
- [iterator](#) **erase** ([iterator](#) \_\_it)

- `iterator erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator find` (`const key_type` &\_\_key)
- `const_iterator find` (`const key_type` &\_\_key) const
- `std::pair< iterator, bool > insert` (`const value_type` &\_\_obj)
- `std::pair< iterator, bool > insert` (`value_type` &&\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`std::pair< iterator, bool > insert` (`_Pair` &&\_\_obj)
- `iterator insert` (`const_iterator` \_\_hint, `const value_type` &\_\_obj)
- `iterator insert` (`const_iterator` \_\_hint, `value_type` &&\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`const_iterator` \_\_hint, `_Pair` &&\_\_obj)
- `void insert` (`std::initializer_list< value_type >` \_\_l)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `float max_load_factor` () const noexcept
- `void max_load_factor` (`float` \_\_f)
- `unordered_map` & `operator=` (`const unordered_map` &)=default
- `unordered_map` & `operator=` (`unordered_map` &&)=default
- `unordered_map` & `operator=` (`initializer_list< value_type >` \_\_l)
- `void swap` (`unordered_map` &\_\_x) noexcept(noexcept(declval< `_Base` & >().swap(\_\_x)))

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_const_local_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- `_Safe_iterator_base` \* `_M_local_iterators`
- unsigned int `_M_version`

#### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- `void _M_invalidate_all` ()
- `void _M_invalidate_all` () const
- `void _M_invalidate_if` (`_Predicate` \_\_pred)
- `void _M_invalidate_local_if` (`_Predicate` \_\_pred)
- `void _M_invalidate_locals` ()
- `void _M_revalidate_singular` ()
- `_Safe_container` & `_M_safe` () noexcept
- `void _M_swap` (`_Safe_unordered_container_base` &\_\_x) noexcept
- `void _M_swap` (`_Safe_sequence_base` &\_\_x) noexcept



#### 5.444.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<std::pair<const _Key, _Tp>>>>
class std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class std::unordered\_map with safety/checking/debug instrumentation.

Definition at line 53 of file debug/unordered\_map.

#### 5.444.2 Member Function Documentation

##### 5.444.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

##### 5.444.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

##### 5.444.2.3 \_M\_get\_mutex()

```
__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.444.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 5.444.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
>::_M_invalidate_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_unordered\_container.tcc.

## 5.444.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
>::_M_invalidate_local_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file safe\_unordered\_container.tcc.

## 5.444.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.444.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    _Safe_unordered_container_base & __x ) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 5.444.2.9 `_M_swap()` [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (  
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.444.3 Member Data Documentation

#### 5.444.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.444.3.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 5.444.3.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.445 std::\_\_debug::unordered\_multimap<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference 1741

### 5.444.3.4 \_M\_local\_iterators

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

### 5.444.3.5 \_M\_version

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

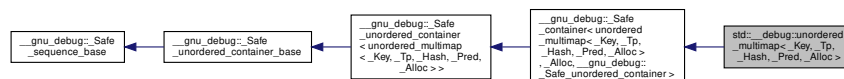
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.445 std::\_\_debug::unordered\_multimap<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inheritance diagram for `std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >`:



### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __gnu_debug::_Safe_iterator<_Base_const_iterator, unordered\_multimap > const_iterator`
- `typedef __gnu_debug::_Safe_local_iterator<_Base_const_local_iterator, unordered\_multimap > const_local_iterator`
- `typedef _Base::hasher hasher`
- `typedef __gnu_debug::_Safe_iterator<_Base_iterator, unordered\_multimap > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef __gnu_debug::_Safe_local_iterator<_Base_local_iterator, unordered\_multimap > local_iterator`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const [unordered\\_multimap](#) &)=default
- **unordered\_multimap** (const [\\_Base](#) &\_\_x)
- **unordered\_multimap** ([unordered\\_multimap](#) &&)=default
- **unordered\_multimap** (const allocator\_type &\_\_a)
- **unordered\_multimap** (const [unordered\\_multimap](#) &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_multimap** ([unordered\\_multimap](#) &&\_\_umap, const allocator\_type &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- [iterator](#) **begin** () noexcept
- const [iterator](#) **begin** () const noexcept
- [local\\_iterator](#) **begin** (size\_type \_\_b)
- const [local\\_iterator](#) **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const [iterator](#) **cbegin** () const noexcept
- const [local\\_iterator](#) **cbegin** (size\_type \_\_b) const
- const [iterator](#) **cend** () const noexcept
- const [local\\_iterator](#) **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
[iterator](#) **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_hint, \_Args &&... \_\_args)
- [iterator](#) **end** () noexcept
- const [iterator](#) **end** () const noexcept
- [local\\_iterator](#) **end** (size\_type \_\_b)
- const [local\\_iterator](#) **end** (size\_type \_\_b) const
- std::pair< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_key)
- std::pair< const [iterator](#), const [iterator](#) > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- [iterator](#) **erase** (const [iterator](#) \_\_it)
- [iterator](#) **erase** ([iterator](#) \_\_it)

- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const value_type &__obj`)
- `iterator insert` (`value_type &&__x`)
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `iterator insert` (`const_iterator __hint`, `value_type &&__x`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`_Pair &&__obj`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (`const_iterator __hint`, `_Pair &&__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `float max_load_factor` () `const noexcept`
- `void max_load_factor` (`float __f`)
- `unordered_multimap & operator=` (`const unordered_multimap &`)=default
- `unordered_multimap & operator=` (`unordered_multimap &&`)=default
- `unordered_multimap & operator=` (`initializer_list< value_type > __l`)
- `void swap` (`unordered_multimap &__x`) `noexcept(noexcept(declval<_Base &>().swap(__x)))`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () `throw` ()
- `void _M_invalidate_all` ()
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_invalidate_local_if` (`_Predicate __pred`)
- `void _M_invalidate_locals` ()
- `void _M_revalidate_singular` ()
- `_Safe_container & _M_safe` () `noexcept`
- `void _M_swap` (`_Safe_unordered_container_base &__x`) `noexcept`
- `void _M_swap` (`_Safe_sequence_base &__x`) `noexcept`

#### 5.445.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<std::pair<const _Key, _Tp>>>>
class std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class std::unordered\_multimap with safety/checking/debug instrumentation.

Definition at line 740 of file debug/unordered\_map.

#### 5.445.2 Member Function Documentation

##### 5.445.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

##### 5.445.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

##### 5.445.2.3 \_M\_get\_mutex()

```
__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.445.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 5.445.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
> ::_M_invalidate_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_unordered\_container.tcc.

## 5.445.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
> ::_M_invalidate_local_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file safe\_unordered\_container.tcc.

## 5.445.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.445.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    _Safe_unordered_container_base & __x ) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.



#### 5.445.2.9 `_M_swap()` [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (  
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.445.3 Member Data Documentation

#### 5.445.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.445.3.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 5.445.3.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 5.445.3.4 \_M\_local\_iterators

```
__Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file safe\_unordered\_base.h.

## 5.445.3.5 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file safe\_base.h.

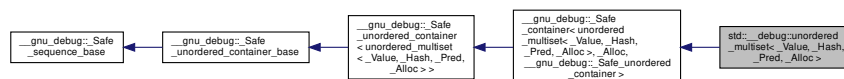
Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_invalidate\_all().

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.446 std::\_\_debug::unordered\_multiset&lt; \_Value, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >:



## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_const\_iterator, unordered\_multiset > **const\_iterator**
- typedef \_\_gnu\_debug::\_Safe\_local\_iterator< \_Base\_const\_local\_iterator, unordered\_multiset > **const\_local\_iterator**
- typedef \_Base::hasher **hasher**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_iterator, unordered\_multiset > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_\_gnu\_debug::\_Safe\_local\_iterator< \_Base\_local\_iterator, unordered\_multiset > **local\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (const [unordered\\_multiset](#) &)=default
- **unordered\_multiset** (const [\\_Base](#) &\_\_x)
- **unordered\_multiset** ([unordered\\_multiset](#) &&)=default
- **unordered\_multiset** (const allocator\_type &\_\_a)
- **unordered\_multiset** (const [unordered\\_multiset](#) &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_multiset** ([unordered\\_multiset](#) &&\_\_uset, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- [iterator begin](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [local\\_iterator begin](#) (size\_type \_\_b)
- [const\\_local\\_iterator begin](#) (size\_type \_\_b) const
- size\_type [bucket\\_size](#) (size\_type \_\_b) const
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_local\\_iterator cbegin](#) (size\_type \_\_b) const
- [const\\_iterator cend](#) () const noexcept
- [const\\_local\\_iterator cend](#) (size\_type \_\_b) const
- void [clear](#) () noexcept
- template<typename... \_Args>  
[iterator emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator emplace\\_hint](#) ([const\\_iterator](#) \_\_hint, \_Args &&... \_\_args)
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [local\\_iterator end](#) (size\_type \_\_b)
- [const\\_local\\_iterator end](#) (size\_type \_\_b) const
- [std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const key\_type &\_\_key)
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > [equal\\_range](#) (const key\_type &\_\_key) const
- size\_type [erase](#) (const key\_type &\_\_key)
- [iterator erase](#) ([const\\_iterator](#) \_\_it)
- [iterator erase](#) ([iterator](#) \_\_it)

- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const value_type &__obj`)
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `iterator insert` (`value_type &&__obj`)
- `iterator insert` (`const_iterator __hint`, `value_type &&__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `float max_load_factor` () `const noexcept`
- `void max_load_factor` (`float __f`)
- `unordered_multiset & operator=` (`const unordered_multiset &`)=`default`
- `unordered_multiset & operator=` (`unordered_multiset &&`)=`default`
- `unordered_multiset & operator=` (`initializer_list< value_type > __l`)
- `void swap` (`unordered_multiset &__x`) `noexcept(noexcept(declval< _Base >().swap(__x)))`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () `throw` ()
- `void _M_invalidate_all` ()
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_invalidate_local_if` (`_Predicate __pred`)
- `void _M_invalidate_locals` ()
- `void _M_revalidate_singular` ()
- `_Safe_container & _M_safe` () `noexcept`
- `void _M_swap` (`_Safe_unordered_container_base &__x`) `noexcept`
- `void _M_swap` (`_Safe_sequence_base &__x`) `noexcept`

#### 5.446.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc =
std::allocator<_Value>>
```

```
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 616 of file `debug/unordered_set`.

## 5.446.2 Member Function Documentation

### 5.446.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

### 5.446.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

### 5.446.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.446.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 5.446.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, _Hash, _Pred, _Alloc >
>::_M_invalidate_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

## 5.446.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, _Hash, _Pred, _Alloc >
>::_M_invalidate_local_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

## 5.446.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.446.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    _Safe_unordered_container_base & __x ) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.446.2.9 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.446.3 Member Data Documentation

#### 5.446.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.446.3.2 `_M_const_local_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators` [inherited]

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 5.446.3.3 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.446.3.4 `_M_local_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

## 5.446.3.5 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file safe\_base.h.

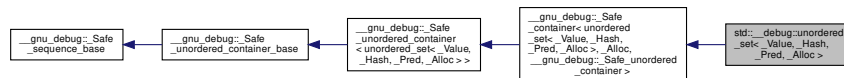
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 5.447 std::\_\_debug::unordered\_set&lt; \_Value, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inheritance diagram for `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_set >` **const\_iterator**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_set >` **const\_local\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, unordered_set >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `__gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_set >` **local\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**



## Public Member Functions

- **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const [unordered\\_set](#) &)=default
- **unordered\_set** (const [\\_Base](#) &\_\_x)
- **unordered\_set** ([unordered\\_set](#) &&)=default
- **unordered\_set** (const allocator\_type &\_\_a)
- **unordered\_set** (const [unordered\\_set](#) &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** ([unordered\\_set](#) &&\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_↵equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_↵type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &↵\_\_a)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- [iterator begin](#) () noexcept
- const [iterator begin](#) () const noexcept
- [local\\_iterator begin](#) (size\_type \_\_b)
- const [local\\_iterator begin](#) (size\_type \_\_b) const
- size\_type [bucket\\_size](#) (size\_type \_\_b) const
- const [iterator cbegin](#) () const noexcept
- const [local\\_iterator cbegin](#) (size\_type \_\_b) const
- const [iterator cend](#) () const noexcept
- const [local\\_iterator cend](#) (size\_type \_\_b) const
- void [clear](#) () noexcept
- template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_hint, \_Args &&... \_\_args)
- [iterator end](#) () noexcept
- const [iterator end](#) () const noexcept
- [local\\_iterator end](#) (size\_type \_\_b)
- const [local\\_iterator end](#) (size\_type \_\_b) const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_key)
- [std::pair](#)< const [iterator](#), const [iterator](#) > **equal\_range** (const key\_type &\_\_key) const
- size\_type [erase](#) (const key\_type &\_\_key)
- [iterator erase](#) (const [iterator](#) \_\_it)
- [iterator erase](#) ([iterator](#) \_\_it)

- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `std::pair< iterator, bool > insert` (`const value_type &__obj`)
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `std::pair< iterator, bool > insert` (`value_type &&__obj`)
- `iterator insert` (`const_iterator __hint`, `value_type &&__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `float max_load_factor` () `const noexcept`
- `void max_load_factor` (`float __f`)
- `unordered_set & operator=` (`const unordered_set &`)=default
- `unordered_set & operator=` (`unordered_set &&`)=default
- `unordered_set & operator=` (`initializer_list< value_type > __l`)
- `void swap` (`unordered_set &__x`) `noexcept(noexcept(declval< _Base & >().swap(__x)))`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () `throw` ()
- `void _M_invalidate_all` ()
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_invalidate_local_if` (`_Predicate __pred`)
- `void _M_invalidate_locals` ()
- `void _M_revalidate_singular` ()
- `_Safe_container & _M_safe` () `noexcept`
- `void _M_swap` (`_Safe_unordered_container_base &__x`) `noexcept`
- `void _M_swap` (`_Safe_sequence_base &__x`) `noexcept`

#### 5.447.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc =
std::allocator<_Value>>
```

```
class std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>
```

Class `std::unordered_set` with safety/checking/debug instrumentation.

Definition at line 53 of file `debug/unordered_set`.

## 5.447.2 Member Function Documentation

### 5.447.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

### 5.447.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

### 5.447.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],  
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

### 5.447.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 5.447.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_set< _Value, _Hash, _Pred, _Alloc > >::↵
_M_invalidate_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

## 5.447.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_set< _Value, _Hash, _Pred, _Alloc > >::↵
_M_invalidate_local_if (
    _Predicate __pred ) [protected], [inherited]
```

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

## 5.447.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.447.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
    _Safe_unordered_container_base & __x ) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.447.2.9 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.447.3 Member Data Documentation

#### 5.447.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.447.3.2 `_M_const_local_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators` [inherited]

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 5.447.3.3 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 5.447.3.4 `_M_local_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

5.447.3.5 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

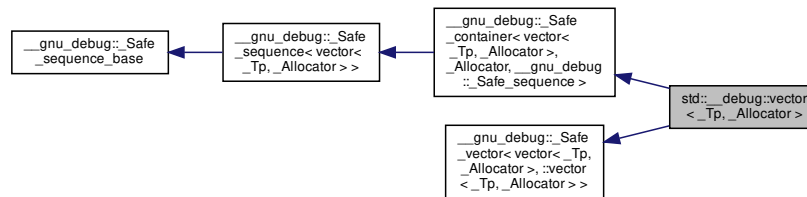
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

5.448 `std::__debug::vector<_Tp, _Allocator>` Class Template Reference

Inheritance diagram for `std::__debug::vector<_Tp, _Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, vector>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, vector>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **vector** (const `_Allocator` &\_\_a) noexcept
- **vector** (size\_type \_\_n, const `_Allocator` &\_\_a=\_Allocator())
- **vector** (size\_type \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=\_Allocator())
- template<class `_InputIterator` , typename = std::RequireInputIter<`_InputIterator`>>  
**vector** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- **vector** (const `vector` &)=default
- **vector** (`vector` &&)=default
- **vector** (const `vector` &\_\_x, const allocator\_type &\_\_a)
- **vector** (`vector` &&\_\_x, const allocator\_type &\_\_a)
- **vector** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- `vector` (const `_Base` &\_\_x)
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_swap** (`_Safe_container` &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- template<typename `_InputIterator` , typename = std::RequireInputIter<`_InputIterator`>>  
void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_u)
- void **assign** (initializer\_list< value\_type > \_\_l)
- reference **back** () noexcept
- const\_reference **back** () const noexcept
- `iterator` **begin** () noexcept
- const `iterator` **begin** () const noexcept
- size\_type **capacity** () const noexcept
- const `iterator` **cbegin** () const noexcept
- const `iterator` **cend** () const noexcept
- void **clear** () noexcept
- const `reverse_iterator` **crbegin** () const noexcept
- const `reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`>  
`iterator` **emplace** (const `iterator` \_\_position, `_Args` &&... \_\_args)
- template<typename... `_Args`>  
void **emplace\_back** (`_Args` &&... \_\_args)
- `iterator` **end** () noexcept
- const `iterator` **end** () const noexcept
- `iterator` **erase** (const `iterator` \_\_position)
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- reference **front** () noexcept
- const\_reference **front** () const noexcept
- `iterator` **insert** (const `iterator` \_\_position, const `_Tp` &\_\_x)
- template<typename `_Up` = `_Tp`>  
\_\_gnu\_cxx::enable\_if<!std::are\_same< `_Up`, bool >::value, `iterator` >::\_\_type **insert** (const `iterator` \_\_position, `_Tp` &&\_\_x)
- `iterator` **insert** (const `iterator` \_\_position, initializer\_list< value\_type > \_\_l)
- `iterator` **insert** (const `iterator` \_\_position, size\_type \_\_n, const `_Tp` &\_\_x)
- template<class `_InputIterator` , typename = std::RequireInputIter<`_InputIterator`>>  
`iterator` **insert** (const `iterator` \_\_position, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `vector` & **operator=** (const `vector` &)=default

- `vector` & `operator=` (`vector` &&)=default
- `vector` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- reference `operator[]` (`size_type` \_\_n) noexcept
- const\_reference `operator[]` (`size_type` \_\_n) const noexcept
- void `pop_back` () noexcept
- void `push_back` (const `_Tp` &\_\_x)
- `template<typename _Up = _Tp>`  
`__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, void >::__type` `push_back` (`_Tp` &&\_\_x)
- `reverse_iterator` `rbegin` () noexcept
- `const_reverse_iterator` `rbegin` () const noexcept
- `reverse_iterator` `rend` () noexcept
- `const_reverse_iterator` `rend` () const noexcept
- void `reserve` (`size_type` \_\_n)
- void `resize` (`size_type` \_\_sz)
- void `resize` (`size_type` \_\_sz, const `_Tp` &\_\_c)
- void `shrink_to_fit` ()
- void `swap` (`vector` &\_\_x) noexcept(*/\*conditional \*/*)

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

#### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::mutex` & `_M_get_mutex` () throw ()
- void `_M_invalidate_all` () const
- bool `_M_requires_reallocation` (`size_type` \_\_elements) const noexcept
- void `_M_revalidate_singular` ()
- `_Safe_container` & `_M_safe` () noexcept
- void `_M_swap` (`_Safe_sequence_base` &\_\_x) noexcept
- void `_M_update_guaranteed_capacity` () noexcept

#### Protected Attributes

- `size_type` `_M_guaranteed_capacity`

#### 5.448.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::vector< _Tp, _Allocator >
```

Class `std::vector` with safety/checking/debug instrumentation.

Definition at line 113 of file `debug/vector`.



## 5.448.2 Constructor & Destructor Documentation

### 5.448.2.1 vector()

```
template<typename _Tp , typename _Allocator = std::allocator<_Tp>>
std::__debug::vector< _Tp, _Allocator >::vector (
    const _Base & __x ) [inline]
```

Construction from a normal-mode vector.

Definition at line 214 of file debug/vector.

## 5.448.3 Member Function Documentation

### 5.448.3.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

### 5.448.3.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

### 5.448.3.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw ( ) [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 5.448.3.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 5.448.3.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< vector< _Tp, _Allocator > >::_M_invalidate_if (
    _Predicate __pred ) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

## 5.448.3.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 5.448.3.7 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
    _Safe_sequence_base & __x ) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 5.448.3.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< vector< _Tp, _Allocator > >::_M_transfer_from_if (
    _Safe_sequence< vector< _Tp, _Allocator > > & __from,
    _Predicate __pred ) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe\_sequence.tcc.

#### 5.448.4 Member Data Documentation

##### 5.448.4.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 5.448.4.2 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 5.448.4.3 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

#### 5.449 `std::detail::BracketMatcher< _TraitsT, __icase, __collate >` Struct Template Reference

##### Public Types

- `typedef _TraitsT::char_class_type _CharClassT`
- `typedef _TransT::CharT _CharT`
- `typedef _TraitsT::string_type _StringT`
- `typedef _TransT::_StrTransT _StrTransT`
- `typedef _RegexTranslator< _TraitsT, __icase, __collate > _TransT`

## Public Member Functions

- **\_BracketMatcher** (bool \_\_is\_non\_matching, const \_TraitsT &\_\_traits)
- void **\_M\_add\_char** (\_CharT \_\_c)
- void **\_M\_add\_character\_class** (const \_StringT &\_\_s, bool \_\_neg)
- \_StringT **\_M\_add\_collate\_element** (const \_StringT &\_\_s)
- void **\_M\_add\_equivalence\_class** (const \_StringT &\_\_s)
- void **\_M\_make\_range** (\_CharT \_\_l, \_CharT \_\_r)
- void **\_M\_ready** ()
- bool **operator()** (\_CharT \_\_ch) const

## 5.449.1 Detailed Description

```
template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher<_TraitsT, __icase, __collate>
```

Matches a character range (bracket expression)

Definition at line 49 of file `regex_compiler.h`.

The documentation for this struct was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

5.450 `std::__detail::_Compiler<_TraitsT>` Class Template Reference

## Public Types

- typedef \_TraitsT::char\_type **\_CharT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef const \_CharT \* **\_IterT**
- typedef \_NFA<\_TraitsT> **\_RegexT**

## Public Member Functions

- **\_Compiler** (\_IterT \_\_b, \_IterT \_\_e, const typename \_TraitsT::locale\_type &\_\_traits, [\\_FlagT](#) \_\_flags)
- [shared\\_ptr](#)< const \_RegexT > **\_M\_get\_nfa** ()

#### 5.450.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_Compiler< _TraitsT >
```

Builds an NFA from an input iterator range.

The `_TraitsT` type should fulfill requirements [28.3].

Definition at line 57 of file `regex_compiler.h`.

The documentation for this class was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

#### 5.451 std::\_\_detail::\_Default\_ranged\_hash Struct Reference

##### 5.451.1 Detailed Description

Default ranged hash function `H`. In principle it should be a function object composed from objects of type `H1` and `H2` such that  $h(k, N) = h2(h1(k), N)$ , but that would mean making extra copies of `h1` and `h2`. So instead we'll just use a tag to tell class template `hashtable` to do that composition.

Definition at line 442 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 5.452 std::\_\_detail::\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, \_\_cache\_hash\_code > Struct Template Reference

##### 5.452.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType, bool __cache_↵
hash_code>
struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
```

Primary class template `_Equal_helper`.

Definition at line 1450 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.453 std::\_\_detail::\_\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, false > Struct Template Reference

### Static Public Member Functions

- static bool **\_S\_equals** (const \_Equal &\_\_eq, const \_ExtractKey &\_\_extract, const \_Key &\_\_k, \_HashCodeType, [\\_Hash\\_node](#)< \_Value, false > \*\_\_n)

#### 5.453.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>
struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >
```

Specialization.

Definition at line 1466 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.454 std::\_\_detail::\_\_Equal\_helper< \_Key, \_Value, \_ExtractKey, \_Equal, \_HashCodeType, true > Struct Template Reference

### Static Public Member Functions

- static bool **\_S\_equals** (const \_Equal &\_\_eq, const \_ExtractKey &\_\_extract, const \_Key &\_\_k, \_HashCodeType \_\_c, [\\_Hash\\_node](#)< \_Value, true > \*\_\_n)

#### 5.454.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>
struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >
```

Specialization.

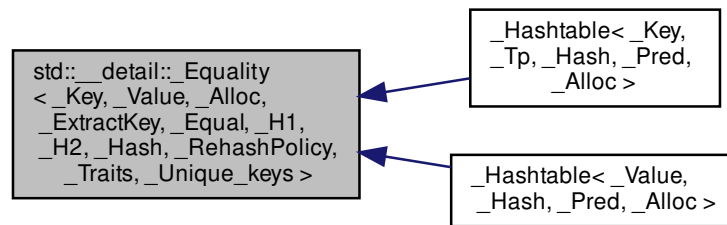
Definition at line 1455 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 5.455 `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



#### 5.455.1 Detailed Description

```

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >

```

Primary class template `_Equality`.

This is for implementing equality comparison for unordered containers, per N3068, by John Lakos and Pablo Halpern. Algorithmically, we follow closely the reference implementations therein.

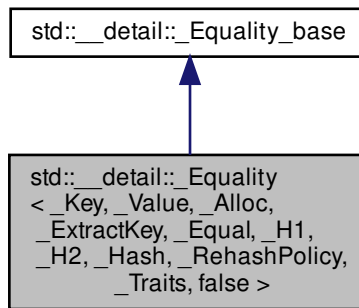
Definition at line 1934 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.456 `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

Inheritance diagram for `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`:



### Public Types

- using `__hashtable` = `_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

### Public Member Functions

- `bool _M_equal` (const `__hashtable` &) const

### Static Protected Member Functions

- `template<typename _Uiterator >`  
`static bool _S_is_permutation` (`_Uiterator, _Uiterator, _Uiterator`)

### 5.456.1 Detailed Description

```

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

```

Specialization.

Definition at line 1979 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



## 5.457 std::\_\_detail::\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true > Struct Template Reference

### Public Types

- using **\_\_hashtable** = [\\_Hashtable](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

### Public Member Functions

- bool **\_M\_equal** (const [\\_\\_hashtable](#) &) const

#### 5.457.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Specialization.

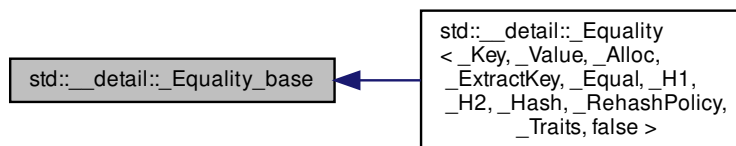
Definition at line 1941 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.458 std::\_\_detail::\_Equality\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Equality\_base:



### Static Protected Member Functions

- template<typename \_Uiterator >  
static bool **\_S\_is\_permutation** (\_Uiterator, \_Uiterator, \_Uiterator)

### 5.458.1 Detailed Description

struct `_Equality_base`.

Common types and functions for class `_Equality`.

Definition at line 1867 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.459 `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >` Class Template Reference

### Public Types

- typedef `iterator_traits<_Bilter >::value_type` `_CharT`
- typedef `_TraitsT::char_class_type` `_ClassT`
- typedef `regex_constants::match_flag_type` `_FlagT`
- typedef `_NFA<_TraitsT >` `_NFAT`
- typedef `basic_regex<_CharT, _TraitsT >` `_RegexT`
- typedef `std::vector<sub_match<_Bilter >, _Alloc >` `_ResultsVec`

### Public Member Functions

- `_Executor` (`_Bilter` \_\_begin, `_Bilter` \_\_end, `_ResultsVec` &\_\_results, const `_RegexT` &\_\_re, `_FlagT` \_\_flags)
- `bool` `_M_match` ()
- `bool` `_M_search` ()
- `bool` `_M_search_from_first` ()

### Public Attributes

- `_Bilter` `_M_begin`
- `_ResultsVec` `_M_cur_results`
- `_Bilter` `_M_current`
- const `_Bilter` `_M_end`
- `_FlagT` `_M_flags`
- `bool` `_M_has_sol`
- const `_NFAT` & `_M_nfa`
- const `_RegexT` & `_M_re`
- `vector<pair<_Bilter, int > >` `_M_rep_count`
- `_ResultsVec` & `_M_results`
- `_State_info<__search_mode, _ResultsVec >` `_M_states`

#### 5.459.1 Detailed Description

```
template<typename _Bilter, typename _Alloc, typename _TraitsT, bool __dfs_mode>
class std::__detail::_Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode >
```

Takes a regex and an input string and does the matching.

The `_Executor` class has two modes: DFS mode and BFS mode, controlled by the template parameter `__dfs_mode`.

Definition at line 60 of file `regex.h`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex\\_executor.h](#)
- [regex\\_executor.tcc](#)

### 5.460 `std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Struct Template Reference

#### 5.460.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_↵
hash_code>
struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Hash_code_base`.

Encapsulates two policy issues that aren't quite orthogonal. (1) the difference between using a ranged hash function and using the combination of a hash function and a range-hashing function. In the former case we don't have such things as hash codes, so we have a dummy type as placeholder. (2) Whether or not we cache hash codes. Caching hash codes is meaningless if we have a ranged hash function.

We also put the key extraction objects here, for convenience. Each specialization derives from one or more of the template parameters to benefit from Ebo. This is important as this type is inherited in some cases by the `_Local_↵iterator_base` type used to implement `local_iterator` and `const_local_iterator`. As with any iterator type we prefer to make it as small as possible.

Primary template is unused except as a hook for specializations.

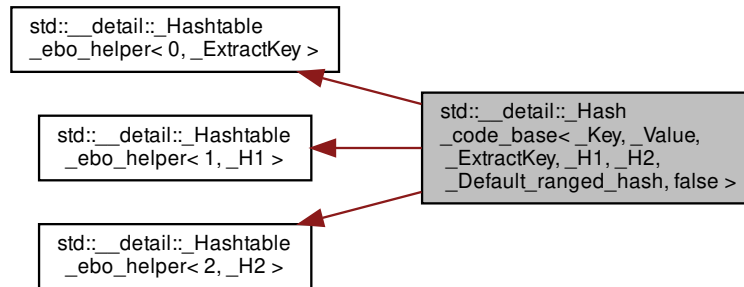
Definition at line 1179 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.461 `std::__detail::_Hash_code_base<_Key,_Value,_ExtractKey,_H1,_H2,_Default_ranged_hash,false>` **Struct Template Reference**

Inheritance diagram for `std::__detail::_Hash_code_base<_Key,_Value,_ExtractKey,_H1,_H2,_Default_ranged_hash,false>`:



### Public Types

- typedef `_H1` **hasher**

### Public Member Functions

- hasher **hash\_function** () const

### Protected Types

- typedef `std::size_t` **\_\_hash\_code**
- typedef `_Hash_node<_Value,false>` **\_\_node\_type**

### Protected Member Functions

- **\_Hash\_code\_base** (const `_ExtractKey` & \_\_ex, const `_H1` & \_\_h1, const `_H2` & \_\_h2, const `_Default_ranged_hash` &)
- `std::size_t` **\_M\_bucket\_index** (const `_Key` &, `__hash_code` \_\_c, `std::size_t` \_\_n) const
- `std::size_t` **\_M\_bucket\_index** (const `__node_type` \* \_\_p, `std::size_t` \_\_n) const noexcept (noexcept (declval< const `_H1` & >>() (declval< const `_Key` & >())) && noexcept (declval< const `_H2` & >>() ((`__hash_code`) 0, (std::size\_t) 0)))
- void **\_M\_copy\_code** (`__node_type` \*, const `__node_type` \*) const
- const `_ExtractKey` & **\_M\_extract** () const
- `_ExtractKey` & **\_M\_extract** ()
- const `_H1` & **\_M\_h1** () const
- `_H1` & **\_M\_h1** ()
- const `_H2` & **\_M\_h2** () const
- `_H2` & **\_M\_h2** ()
- `__hash_code` **\_M\_hash\_code** (const `_Key` & \_\_k) const
- void **\_M\_store\_code** (`__node_type` \*, `__hash_code`) const
- void **\_M\_swap** (`_Hash_code_base` & \_\_x)

## Friends

- struct `_Local_iterator_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Default_ranged_hash`, `false` >

## 5.461.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>
struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >
```

Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.

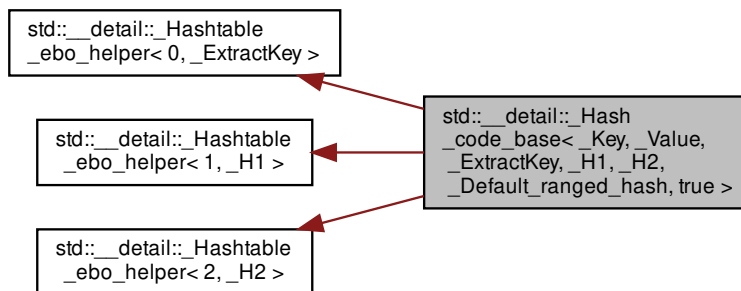
Definition at line 1262 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.462 `std::__detail::_Hash_code_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Default_ranged_hash`, `true` > Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Default_ranged_hash`, `true` >:



## Public Types

- typedef `_H1` **hasher**

## Public Member Functions

- hasher **hash\_function** () const

## Protected Types

- typedef std::size\_t **\_\_hash\_code**
- typedef [\\_Hash\\_node](#)< \_Value, true > **\_\_node\_type**

## Protected Member Functions

- **\_Hash\_code\_base** (const \_ExtractKey &\_\_ex, const \_H1 &\_\_h1, const \_H2 &\_\_h2, const [\\_Default\\_ranged\\_hash](#) &)
- std::size\_t **\_M\_bucket\_index** (const \_Key &, \_\_hash\_code \_\_c, std::size\_t \_\_n) const
- std::size\_t **\_M\_bucket\_index** (const [\\_\\_node\\_type](#) \*\_\_p, std::size\_t \_\_n) const noexcept(noexcept(declval<const \_H2 &>().\_\_hash\_code() 0,(std::size\_t) 0)))
- void **\_M\_copy\_code** ([\\_\\_node\\_type](#) \*\_\_to, const [\\_\\_node\\_type](#) \*\_\_from) const
- const \_ExtractKey & **\_M\_extract** () const
- \_ExtractKey & **\_M\_extract** ()
- const \_H1 & **\_M\_h1** () const
- \_H1 & **\_M\_h1** ()
- const \_H2 & **\_M\_h2** () const
- \_H2 & **\_M\_h2** ()
- \_\_hash\_code **\_M\_hash\_code** (const \_Key &\_\_k) const
- void **\_M\_store\_code** ([\\_\\_node\\_type](#) \*\_\_n, \_\_hash\_code \_\_c) const
- void **\_M\_swap** ([\\_Hash\\_code\\_base](#) &\_\_x)

## Friends

- struct **\_Local\_iterator\_base**< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, true >

## 5.462.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>
struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >
```

Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.

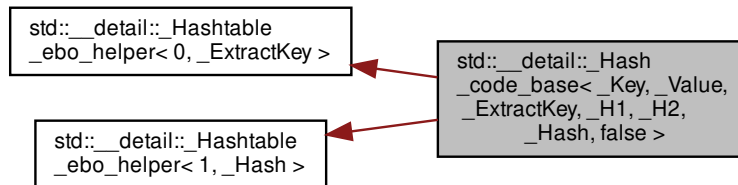
Definition at line 1356 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 5.463 `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`:



#### Protected Types

- `typedef void * __hash_code`
- `typedef \_Hash\_node<_Value, false > __node_type`

#### Protected Member Functions

- `_Hash_code_base` (`const _ExtractKey &__ex, const _H1 &, const _H2 &, const _Hash &__h`)
- `std::size_t _M_bucket_index` (`const _Key &__k, __hash_code, std::size_t __n`) `const`
- `std::size_t _M_bucket_index` (`const \_\_node\_type * __p, std::size_t __n`) `const noexcept`(`noexcept(declval<const _Hash &>())(declval<const _Key &>()),(std::size_t) 0))`)
- `void _M_copy_code` (`\_\_node\_type *, const \_\_node\_type *`) `const`
- `const _ExtractKey & _M_extract` () `const`
- `_ExtractKey & _M_extract` ()
- `__hash_code _M_hash_code` (`const _Key &__key`) `const`
- `const _Hash & _M_ranged_hash` () `const`
- `_Hash & _M_ranged_hash` ()
- `void _M_store_code` (`\_\_node\_type *, __hash_code`) `const`
- `void _M_swap` (`\_Hash\_code\_base &__x`)

#### 5.463.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
struct std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

Definition at line 1185 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.464 std::\_\_detail::\_Hash\_node< \_Value, \_Cache\_hash\_code > Struct Template Reference

### 5.464.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>
struct std::__detail::_Hash_node< _Value, _Cache_hash_code >
```

Primary template struct \_Hash\_node.

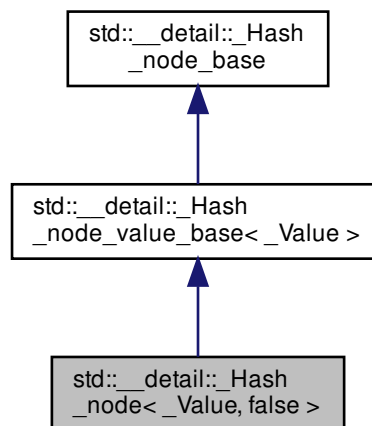
Definition at line 257 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.465 std::\_\_detail::\_Hash\_node< \_Value, false > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node< \_Value, false >:



### Public Types

- typedef \_Value **value\_type**



## Public Member Functions

- [\\_Hash\\_node](#) \* **\_M\_next** () const noexcept
- [\\_Value](#) & **\_M\_v** () noexcept
- const [\\_Value](#) & **\_M\_v** () const noexcept
- [\\_Value](#) \* **\_M\_valptr** () noexcept
- const [\\_Value](#) \* **\_M\_valptr** () const noexcept

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* **\_M\_nxt**
- `__gnu_cxx::__aligned_buffer< \_Value >` **\_M\_storage**

## 5.465.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node< _Value, false >
```

Specialization for nodes without caches, struct `_Hash_node`.

Base class is `__detail::_Hash_node_value_base`.

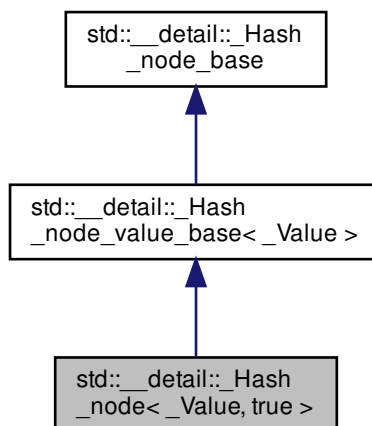
Definition at line 280 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

5.466 `std::__detail::_Hash_node< _Value, true >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node< _Value, true >`:



## Public Types

- typedef \_Value **value\_type**

## Public Member Functions

- [\\_Hash\\_node](#) \* **\_M\_next** () const noexcept
- \_Value & **\_M\_v** () noexcept
- const \_Value & **\_M\_v** () const noexcept
- \_Value \* **\_M\_valptr** () noexcept
- const \_Value \* **\_M\_valptr** () const noexcept

## Public Attributes

- std::size\_t **\_M\_hash\_code**
- [\\_Hash\\_node\\_base](#) \* **\_M\_nxt**
- \_\_gnu\_cxx::\_\_aligned\_buffer< \_Value > **\_M\_storage**

## 5.466.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node< _Value, true >
```

Specialization for nodes with caches, struct \_Hash\_node.

Base class is \_\_detail::\_Hash\_node\_value\_base.

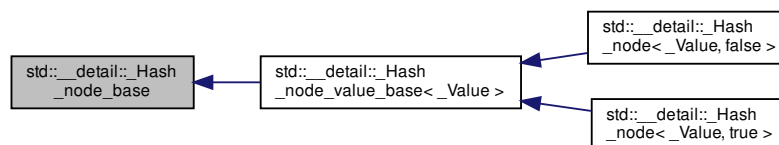
Definition at line 265 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.467 std::\_\_detail::\_Hash\_node\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node\_base:



## Public Member Functions

- `_Hash_node_base` (`_Hash_node_base *__next`) noexcept

## Public Attributes

- `_Hash_node_base * _M_nxt`

## 5.467.1 Detailed Description

struct `_Hash_node_base`

Nodes, used to wrap elements stored in the hash table. A policy template parameter of class template `_Hashtable` controls whether nodes also store a hash code. In some cases (e.g. strings) this may be a performance win.

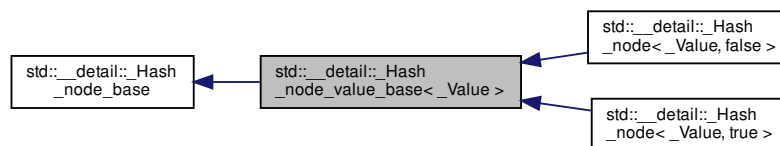
Definition at line 215 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

5.468 `std::__detail::_Hash_node_value_base< _Value >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node_value_base< _Value >`:



## Public Types

- `typedef _Value value_type`

## Public Member Functions

- `_Value & _M_v ()` noexcept
- `const _Value & _M_v ()` const noexcept
- `_Value * _M_valptr ()` noexcept
- `const _Value * _M_valptr ()` const noexcept

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* [\\_M\\_nxt](#)
- [\\_\\_gnu\\_cxx::\\_\\_aligned\\_buffer](#)< [\\_Value](#) > [\\_M\\_storage](#)

## 5.468.1 Detailed Description

```
template<typename _Value>
struct std::__detail::__Hash_node_value_base< _Value >
```

```
struct _Hash_node_value_base
```

Node type with the value to store.

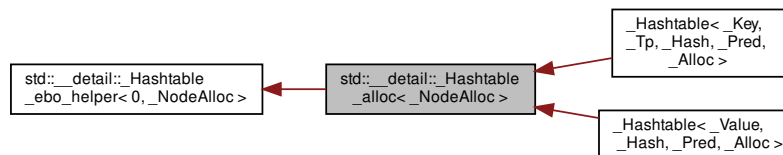
Definition at line 230 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.469 std::\_\_detail::\_\_Hashtable\_alloc&lt; \_NodeAlloc &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_\_Hashtable\_alloc< \_NodeAlloc >:



## Public Types

- using `__bucket_alloc_traits` = [std::allocator\\_traits](#)< `__bucket_alloc_type` >
- using `__bucket_alloc_type` = `__alloc_rebind`< `__node_alloc_type`, [\\_\\_bucket\\_type](#) >
- using `__bucket_type` = `__node_base` \*
- using `__node_alloc_traits` = [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< `__node_alloc_type` >
- using `__node_alloc_type` = `_NodeAlloc`
- using `__node_base` = [\\_\\_detail::\\_\\_Hash\\_node\\_base](#)
- using `__node_type` = `typename _NodeAlloc::value_type`
- using `__value_alloc_traits` = `typename __node_alloc_traits::template rebind_traits< typename __node_type::value_type >`

## Public Member Functions

- `_Hashtable_alloc` (const `_Hashtable_alloc` &)=default
- `_Hashtable_alloc` (`_Hashtable_alloc` &&)=default
- `template<typename _Alloc >`  
`_Hashtable_alloc` (`_Alloc` &&`_a`)
- `_bucket_type` \* `_M_allocate_buckets` (std::size\_t `_n`)
- `template<typename... _Args>`  
`_node_type` \* `_M_allocate_node` (`_Args` &&... `_args`)
- `void` `_M_deallocate_buckets` (`_bucket_type` \*, std::size\_t `_n`)
- `void` `_M_deallocate_node` (`_node_type` \* `_n`)
- `void` `_M_deallocate_nodes` (`_node_type` \* `_n`)
- `_node_alloc_type` & `_M_node_allocator` ()
- `const` `_node_alloc_type` & `_M_node_allocator` () `const`

## 5.469.1 Detailed Description

```
template<typename _NodeAlloc>
struct std::__detail::_Hashtable_alloc< _NodeAlloc >
```

This type deals with all allocation and keeps an allocator instance through inheritance to benefit from EBO when possible.

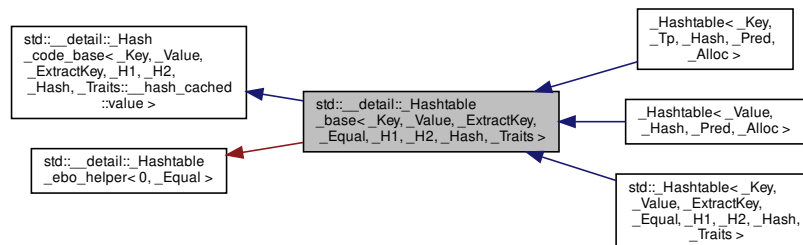
Definition at line 98 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.470 `std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`:



## Public Types

- using `__constant_iterators` = `typename __traits_type::__constant_iterators`
- using `__hash_cached` = `typename __traits_type::__hash_cached`
- using `__hash_code` = `typename __hash_code_base::__hash_code`
- using `__hash_code_base` = `_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __hash_↵cached::value>`
- using `__ireturn_type` = `typename std::conditional<__unique_keys::value, std::pair< iterator, bool>, iterator>::type`
- using `__node_type` = `typename __hash_code_base::__node_type`
- using `__traits_type` = `_Traits`
- using `__unique_keys` = `typename __traits_type::__unique_keys`
- using `const_iterator` = `__detail::__Node_const_iterator<value_type, __constant_iterators::value, __hash_↵cached::value>`
- using `const_local_iterator` = `__detail::__Local_const_iterator<key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __constant_iterators::value, __hash_cached::value>`
- `typedef std::ptrdiff_t difference_type`
- using `iterator` = `__detail::__Node_iterator<value_type, __constant_iterators::value, __hash_cached::value>`
- `typedef _Equal key_equal`
- `typedef _Key key_type`
- using `local_iterator` = `__detail::__Local_iterator<key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __↵constant_iterators::value, __hash_cached::value>`
- `typedef std::size_t size_type`
- `typedef _Value value_type`

## Protected Member Functions

- `_Hashtable_base` (`const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const _Hash &__hash, const _Equal &__eq`)
- `const _Equal &_M_eq` () const
- `_Equal &_M_eq` ()
- `bool _M_equals` (`const _Key &__k, __hash_code __c, __node_type *__n`) const
- `void _M_swap` (`_Hashtable_base &__x`)

## 5.470.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash,
typename _Traits>
```

```
struct std::__detail::__Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits>
```

Primary class template `_Hashtable_base`.

Helper class adding management of `_Equal` functor to `_Hash_code_base` type.

Base class templates are:

- `__detail::__Hash_code_base`
- `__detail::__Hashtable_ebo_helper`

Definition at line 58 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.471 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, __use_ebo >` Struct Template Reference

### 5.471.1 Detailed Description

```
template<int _Nm, typename _Tp, bool __use_ebo = !__is_final(_Tp) && __is_empty(_Tp)>
struct std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, __use_ebo >
```

Primary class template `_Hashtable_ebo_helper`.

Helper class using EBO when it is not forbidden (the type is not final) and when it is worth it (the type is empty.)

Definition at line 1099 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.472 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false >` Struct Template Reference

### Public Member Functions

- `template<typename _OtherTp >`  
`_Hashtable_ebo_helper (_OtherTp && __tp)`

### Static Public Member Functions

- `static const _Tp & _S_cget (const \_Hashtable\_ebo\_helper &__eboh)`
- `static _Tp & _S_get (\_Hashtable\_ebo\_helper &__eboh)`

### 5.472.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false >
```

Specialization not using EBO.

Definition at line 1124 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.473 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, true >` Struct Template Reference

Inherits `_Tp`.

#### Public Member Functions

- `template<typename _OtherTp >`  
`__Hashtable_ebo_helper ( _OtherTp &&__tp)`

#### Static Public Member Functions

- `static const _Tp & __S_cget (const __Hashtable_ebo_helper &__eboh)`
- `static _Tp & __S_get ( __Hashtable_ebo_helper &__eboh)`

##### 5.473.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >
```

Specialization using EBO.

Definition at line 1103 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 5.474 `std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >` Struct Template Reference

##### Public Types

- using `__constant_iterators` = `__bool_constant< _Constant_iterators >`
- using `__hash_cached` = `__bool_constant< _Cache_hash_code >`
- using `__unique_keys` = `__bool_constant< _Unique_keys >`

##### 5.474.1 Detailed Description

```
template<bool _Cache_hash_code, bool _Constant_iterators, bool _Unique_keys>
struct std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
```

`struct _Hashtable_traits`

Important traits for hash tables.



## Template Parameters

<code>_Cache_hash_code</code>	Boolean value. True if the value of the hash function is stored along with the value. This is a time-space tradeoff. Storing it may improve lookup speed by reducing the number of times we need to call the <code>_Equal</code> function.
<code>_Constant_iterators</code>	Boolean value. True if <code>iterator</code> and <code>const_iterator</code> are both constant iterator types. This is true for <code>unordered_set</code> and <code>unordered_multiset</code> , false for <code>unordered_map</code> and <code>unordered_multimap</code> .
<code>_Unique_keys</code>	Boolean value. True if the return value of <code>_Hashtable::count(k)</code> is always at most one, false if it may be an arbitrary number. This is true for <code>unordered_set</code> and <code>unordered_map</code> , false for <code>unordered_multiset</code> and <code>unordered_multimap</code> .

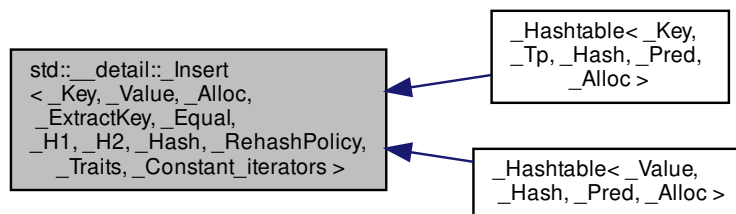
Definition at line 200 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 5.475 `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators>` Struct Template Reference

Inheritance diagram for `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators>`:



##### 5.475.1 Detailed Description

```

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Constant_iterators = _Traits::__constant_iterators::value>
struct std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators>

```

Primary class template `_Insert`.

Defines `insert` member functions that depend on `_Hashtable` policies, via partial specializations.

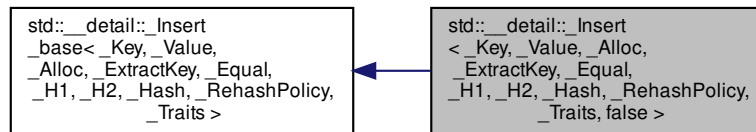
Definition at line 929 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.476 std::\_\_detail::Insert<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false > Struct Template Reference

Inheritance diagram for std::\_\_detail::Insert<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >:



### Public Types

- using **\_\_base\_type** = [Insert\\_base](#)<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using **\_\_hashtable** = typename [\\_\\_base\\_type::\\_\\_hashtable](#)
- using **\_\_ireturn\_type** = typename [\\_\\_base\\_type::\\_\\_ireturn\\_type](#)
- template<typename \_Pair >  
using **\_\_is\_cons** = [std::is\\_constructible](#)< value\_type, \_Pair && >
- using **\_\_unique\_keys** = typename [\\_\\_base\\_type::\\_\\_unique\\_keys](#)
- template<typename \_Pair >  
using **\_\_IFcons** = [std::enable\\_if](#)< [\\_\\_is\\_cons](#)< \_Pair >::value >
- template<typename \_Pair >  
using **\_\_IFcons\_p** = typename [\\_\\_IFcons](#)< \_Pair >::type
- using **const\_iterator** = typename [\\_\\_base\\_type::const\\_iterator](#)
- using **iterator** = typename [\\_\\_base\\_type::iterator](#)
- using **value\_type** = typename [\\_\\_base\\_type::value\\_type](#)

### Public Member Functions

- [\\_\\_ireturn\\_type](#) **insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_v)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_v)
- [\\_\\_ireturn\\_type](#) **insert** (const value\_type &\_\_v)
- template<typename \_Pair, typename = [\\_\\_IFcons\\_p](#)< \_Pair >>  
[\\_\\_ireturn\\_type](#) **insert** (\_Pair &&\_\_v)
- template<typename \_Pair, typename = [\\_\\_IFcons\\_p](#)< \_Pair >>  
iterator **insert** (const\_iterator \_\_hint, \_Pair &&\_\_v)

## Protected Types

- using **\_\_hashtable\_base** = [\\_Hashtable\\_base](#)< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using **\_\_node\_alloc\_type** = [\\_\\_alloc\\_rebind](#)< \_Alloc, [\\_\\_node\\_type](#) >
- using **\_\_node\_gen\_type** = [\\_AllocNode](#)< \_\_node\_alloc\_type >
- using **\_\_node\_type** = [\\_Hash\\_node](#)< \_Value, \_Traits::\_\_hash\_cached::value >
- using **size\_type** = typename \_\_hashtable\_base::size\_type

## Protected Member Functions

- [\\_\\_hashtable](#) & **\_M\_conjure\_hashtable** ()
- template<typename \_InputIterator, typename \_NodeGetter >  
void **\_M\_insert\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_NodeGetter &, [true\\_type](#))
- template<typename \_InputIterator, typename \_NodeGetter >  
void **\_M\_insert\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_NodeGetter &, [false\\_type](#))

## 5.476.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

Specialization.

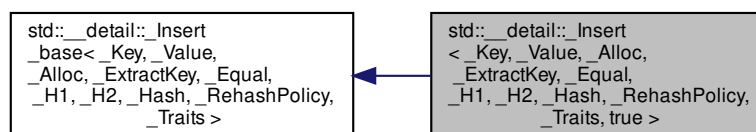
Definition at line 983 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.477 std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true > Struct Template Reference

Inheritance diagram for std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >:



## Public Types

- using **\_\_base\_type** = [\\_Insert\\_base](#)<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using **\_\_hashtable** = typename [\\_\\_base\\_type::\\_\\_hashtable](#)
- using **\_\_hashtable\_base** = [\\_Hashtable\\_base](#)<\_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using **\_\_ireturn\_type** = typename [\\_\\_hashtable\\_base::\\_\\_ireturn\\_type](#)
- using **\_\_node\_gen\_type** = typename [\\_\\_base\\_type::\\_\\_node\\_gen\\_type](#)
- using **\_\_unique\_keys** = typename [\\_\\_base\\_type::\\_\\_unique\\_keys](#)
- using **const\_iterator** = typename [\\_\\_base\\_type::const\\_iterator](#)
- using **iterator** = typename [\\_\\_base\\_type::iterator](#)
- using **value\_type** = typename [\\_\\_base\\_type::value\\_type](#)

## Public Member Functions

- [\\_\\_ireturn\\_type](#) **insert** (const [value\\_type](#) &\_\_v)
- iterator **insert** (const\_iterator \_\_hint, const [value\\_type](#) &\_\_v)
- void **insert** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- iterator **insert** (const\_iterator \_\_hint, const [value\\_type](#) &\_\_v)
- [\\_\\_ireturn\\_type](#) **insert** (const [value\\_type](#) &\_\_v)
- [\\_\\_ireturn\\_type](#) **insert** ([value\\_type](#) &&\_\_v)
- iterator **insert** (const\_iterator \_\_hint, [value\\_type](#) &&\_\_v)

## Protected Types

- using **\_\_node\_alloc\_type** = [\\_alloc\\_rebind](#)<\_Alloc, [\\_\\_node\\_type](#) >
- using **\_\_node\_type** = [\\_Hash\\_node](#)<\_Value, \_Traits::\_\_hash\_cached::value >
- using **size\_type** = typename [\\_\\_hashtable\\_base::size\\_type](#)

## Protected Member Functions

- [\\_\\_hashtable](#) & **M\_conjure\_hashtable** ()
- template<typename \_InputIterator, typename \_NodeGetter >  
void **M\_insert\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_NodeGetter &, [true\\_type](#))
- template<typename \_InputIterator, typename \_NodeGetter >  
void **M\_insert\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_NodeGetter &, [false\\_type](#))

## 5.477.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Specialization.

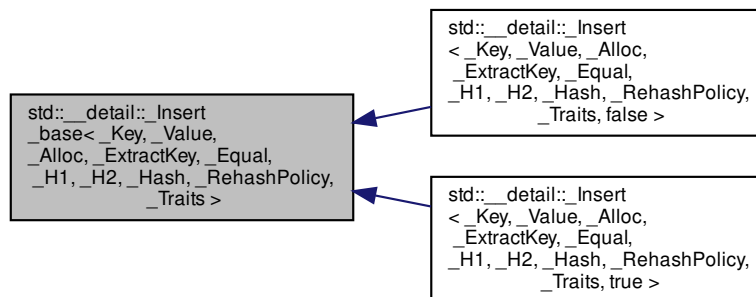
Definition at line 936 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.478 std::\_\_detail::\_\_Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_\_Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >:



### Public Member Functions

- `__ireturn_type` **insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_v)
- void **insert** (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)

## Protected Types

- using `__hashtable` = `_Hashtable`< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using `__hashtable_base` = `_Hashtable_base`< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using `__ireturn_type` = `typename __hashtable_base::__ireturn_type`
- using `__node_alloc_type` = `_alloc_rebind`< \_Alloc, `__node_type` >
- using `__node_gen_type` = `_AllocNode`< `__node_alloc_type` >
- using `__node_type` = `_Hash_node`< \_Value, \_Traits::\_\_hash\_cached::value >
- using `__unique_keys` = `typename __hashtable_base::__unique_keys`
- using `const_iterator` = `typename __hashtable_base::const_iterator`
- using `iterator` = `typename __hashtable_base::iterator`
- using `size_type` = `typename __hashtable_base::size_type`
- using `value_type` = `typename __hashtable_base::value_type`

## Protected Member Functions

- `__hashtable & _M_conjure_hashtable ()`
- `template<typename _InputIterator, typename _NodeGetter >`  
`void _M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &, true_type)`
- `template<typename _InputIterator, typename _NodeGetter >`  
`void _M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &, false_type)`

## 5.478.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Insert_base`.

Defines `insert` member functions appropriate to all `_Hashtables`.

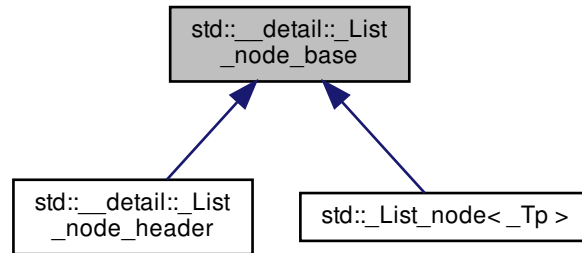
Definition at line 792 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 5.479 std::\_\_detail::\_List\_node\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_List\_node\_base:



#### Public Member Functions

- void **\_M\_hook** ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept

#### Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

#### Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 5.479.1 Detailed Description

Common part of a node in the list.

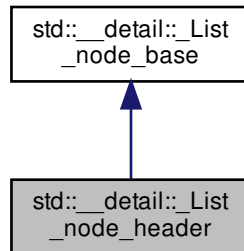
Definition at line 80 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.480 std::\_\_detail::\_List\_node\_header Struct Reference

Inheritance diagram for std::\_\_detail::\_List\_node\_header:



## Public Member Functions

- [\\_List\\_node\\_header](#) ([\\_List\\_node\\_header](#) &&\_\_x) noexcept
- void [\\_M\\_hook](#) ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void [\\_M\\_init](#) () noexcept
- void [\\_M\\_move\\_nodes](#) ([\\_List\\_node\\_header](#) &&\_\_x)
- void [\\_M\\_reverse](#) () noexcept
- void [\\_M\\_transfer](#) ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void [\\_M\\_unhook](#) () noexcept

## Static Public Member Functions

- static void [swap](#) ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

## Public Attributes

- [\\_List\\_node\\_base](#) \* [\\_M\\_next](#)
- [\\_List\\_node\\_base](#) \* [\\_M\\_prev](#)

## 5.480.1 Detailed Description

The list node header.

Definition at line 103 of file [stl\\_list.h](#).

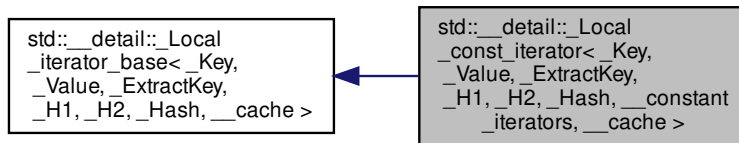
The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)



## 5.481 `std::__detail::_Local_const_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_const_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

### Public Member Functions

- **\_Local\_const\_iterator** (`const __hash_code_base &_base, \_Hash\_node<_Value, __cache > *__p, std::size_t __bkt, std::size_t __bkt_count`)
- **\_Local\_const\_iterator** (`const \_Local\_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > &__x`)
- reference **operator\*** () const
- [\\_Local\\_const\\_iterator](#) & **operator++** ()
- [\\_Local\\_const\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

#### 5.481.1 Detailed Description

```

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>
struct std::__detail::_Local_const_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

```

local const\_iterators

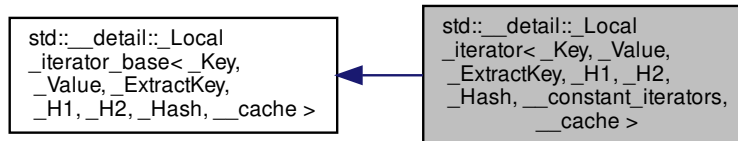
Definition at line 1712 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

5.482 `std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `std::conditional< __constant_iterators, const _Value *, _Value * >::type` **pointer**
- typedef `std::conditional< __constant_iterators, const _Value &, _Value & >::type` **reference**
- typedef `_Value` **value\_type**

#### Public Member Functions

- **\_Local\_iterator** (`const __hash_code_base &__base, \_Hash\_node< _Value, __cache > *__p, std::size_t __bkt, std::size_t __bkt_count`)
- reference **operator\*** () const
- [\\_Local\\_iterator](#) & **operator++** ()
- [\\_Local\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

#### 5.482.1 Detailed Description

```

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>
struct std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

```

local iterators

Definition at line 1657 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.483 `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Struct Template Reference

### 5.483.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code <→
hash_code>
struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Local_iterator_base`.

Base class for local iterators, used to iterate within a bucket but not between buckets.

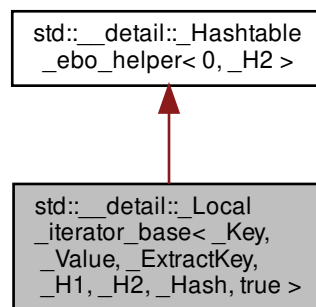
Definition at line 1154 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.484 `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`:



### Public Member Functions

- `const void * _M_curr () const`
- `std::size_t _M_get_bucket () const`

#### Protected Types

- using `__base_type` = `_Hashtable_ebo_helper< 0, _H2 >`
- using `__hash_code_base` = `_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`

#### Protected Member Functions

- `_Local_iterator_base` (const `_hash_code_base` &\_\_base, `_Hash_node< _Value, true > * __p`, `std::size_t __bkt`, `std::size_t __bkt_count`)
- void `_M_incr` ()

#### Protected Attributes

- `std::size_t _M_bucket`
- `std::size_t _M_bucket_count`
- `_Hash_node< _Value, true > * _M_cur`

#### 5.484.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
struct std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >
```

Partial specialization used when nodes contain a cached hash code.

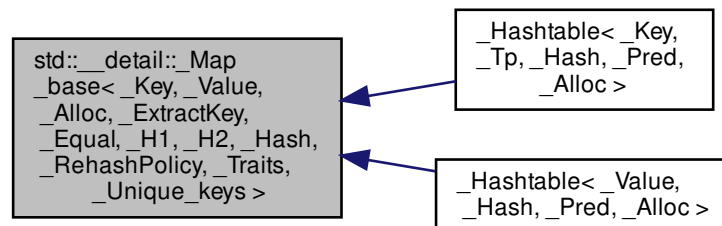
Definition at line 1478 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 5.485 `std::__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



#### 5.485.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
```

Primary class template `_Map_base`.

If the hashtable has a value type of the form `pair<T1, T2>` and a key extraction policy (`_ExtractKey`) that returns the first part of the pair, the hashtable gets a `mapped_type` typedef. If it satisfies those criteria and also has unique keys, then it also gets an `operator[]`.

Definition at line 643 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

**5.486** `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

#### Public Types

- using **mapped\_type** = typename `std::tuple_element< 1, _Pair >::type`

#### 5.486.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits>
struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

Partial specialization, `__unique_keys` set to false.

Definition at line 649 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

**5.487** `std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

#### Public Types

- using **iterator** = typename `__hashtable_base::iterator`
- using **key\_type** = typename `__hashtable_base::key_type`
- using **mapped\_type** = typename `std::tuple_element< 1, _Pair >::type`

### Public Member Functions

- mapped\_type & **at** (const key\_type &\_\_k)
- const mapped\_type & **at** (const key\_type &\_\_k) const
- mapped\_type & **operator[]** (const key\_type &\_\_k)
- mapped\_type & **operator[]** (key\_type &&\_\_k)

#### 5.487.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-  
name _RehashPolicy, typename _Traits>  
struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Partial specialization, \_\_unique\_keys set to true.

Definition at line 659 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.488 std::\_\_detail::\_Mask\_range\_hashing Struct Reference

### Public Types

- typedef std::size\_t **first\_argument\_type**
- typedef std::size\_t **result\_type**
- typedef std::size\_t **second\_argument\_type**

### Public Member Functions

- result\_type **operator()** (first\_argument\_type \_\_num, second\_argument\_type \_\_den) const noexcept

#### 5.488.1 Detailed Description

Range hashing function assuming that second arg is a power of 2.

Definition at line 495 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.489 std::\_\_detail::\_Mod\_range\_hashing Struct Reference

### Public Types

- typedef std::size\_t **first\_argument\_type**
- typedef std::size\_t **result\_type**
- typedef std::size\_t **second\_argument\_type**

### Public Member Functions

- result\_type **operator()** (first\_argument\_type \_\_num, second\_argument\_type \_\_den) const noexcept

### 5.489.1 Detailed Description

Default range hashing function: use division to fold a large number into the range [0, N).

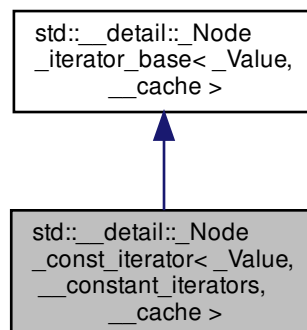
Definition at line 425 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.490 std::\_\_detail::\_Node\_const\_iterator< \_Value, \_\_constant\_iterators, \_\_cache > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Node\_const\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >:



## Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Node\_const\_iterator** (`_node_type *__p`) noexcept
- **\_Node\_const\_iterator** (`const \_Node\_iterator<_Value, __constant_iterators, __cache > &__x`) noexcept
- `void _M_incr ()` noexcept
- `reference operator* ()` const noexcept
- `\_Node\_const\_iterator & operator++ ()` noexcept
- `\_Node\_const\_iterator operator++ (int)` noexcept
- `pointer operator-> ()` const noexcept

## Public Attributes

- `__node_type * _M_cur`

### 5.490.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>
struct std::__detail::_Node_const_iterator<_Value, __constant_iterators, __cache >
```

Node `const_iterators`, used to iterate through all the hashtable.

Definition at line 370 of file `hashtable_policy.h`.

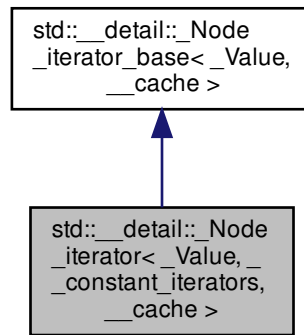
The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



## 5.491 `std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >`:



### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- using **pointer** = typename `std::conditional< __constant_iterators, const _Value *, _Value * >::type`
- using **reference** = typename `std::conditional< __constant_iterators, const _Value &, _Value & >::type`
- typedef `_Value` **value\_type**

### Public Member Functions

- **\_Node\_iterator** (`__node_type * __p`) noexcept
- void **\_M\_incr** () noexcept
- reference **operator\*** () const noexcept
- `_Node_iterator` & **operator++** () noexcept
- `_Node_iterator` **operator++** (int) noexcept
- pointer **operator->** () const noexcept

### Public Attributes

- `__node_type *` **\_M\_cur**

#### 5.491.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>
struct std::__detail::_Node_iterator<_Value, __constant_iterators, __cache>
```

Node iterators, used to iterate through all the hashtable.

Definition at line 319 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 5.492 `std::__detail::_Node_iterator_base<_Value, _Cache_hash_code>` Struct Template Reference

#### Public Types

- using `__node_type` = [\\_Hash\\_node](#)<\_Value, \_Cache\_hash\_code>

#### Public Member Functions

- `_Node_iterator_base` ([\\_\\_node\\_type](#) \*\_\_p) noexcept
- `void _M_incr` () noexcept

#### Public Attributes

- [\\_\\_node\\_type](#) \* `_M_cur`

#### 5.492.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>
struct std::__detail::_Node_iterator_base<_Value, _Cache_hash_code>
```

Base class for node iterators.

Definition at line 289 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 5.493 `std::__detail::_Power2_rehash_policy` Struct Reference

#### Public Types

- using `__has_load_factor` = `std::true_type`
- typedef `std::size_t` `_State`

#### Public Member Functions

- `_Power2_rehash_policy` (float `__z`=1.0) noexcept
- `std::size_t` `_M_bkt_for_elements` (std::size\_t `__n`) const noexcept
- `std::pair`< bool, std::size\_t > `_M_need_rehash` (std::size\_t `__n_bkt`, std::size\_t `__n_elt`, std::size\_t `__n_ins`) noexcept
- `std::size_t` `_M_next_bkt` (std::size\_t `__n`) noexcept
- void `_M_reset` () noexcept
- void `_M_reset` (\_State `__state`) noexcept
- \_State `_M_state` () const noexcept
- float `max_load_factor` () const noexcept

#### Public Attributes

- float `_M_max_load_factor`
- `std::size_t` `_M_next_resize`

#### Static Public Attributes

- static const `std::size_t` `_S_growth_factor`

#### 5.493.1 Detailed Description

Rehash policy providing power of 2 bucket numbers. Avoids modulo operations.

Definition at line 532 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 5.494 `std::__detail::_Prime_rehash_policy` Struct Reference

#### Public Types

- using `__has_load_factor` = `std::true_type`
- typedef `std::size_t` `_State`

## Public Member Functions

- `_Prime_rehash_policy` (float \_\_z=1.0) noexcept
- `std::size_t _M_bkt_for_elements` (std::size\_t \_\_n) const
- `std::pair< bool, std::size_t > _M_need_rehash` (std::size\_t \_\_n\_bkt, std::size\_t \_\_n\_elt, std::size\_t \_\_n\_ins) const
- `std::size_t _M_next_bkt` (std::size\_t \_\_n) const
- `void _M_reset` () noexcept
- `void _M_reset` (\_State \_\_state)
- `_State _M_state` () const
- `float max_load_factor` () const noexcept

## Public Attributes

- `float _M_max_load_factor`
- `std::size_t _M_next_resize`

## Static Public Attributes

- `static const std::size_t _S_growth_factor`

## 5.494.1 Detailed Description

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

Definition at line 446 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

5.495 `std::__detail::_Quoted_string<_String, _CharT>` Struct Template Reference

## Public Member Functions

- `_Quoted_string` (\_String \_\_str, \_CharT \_\_del, \_CharT \_\_esc)
- `_Quoted_string & operator=` (\_Quoted\_string &)=delete

## Public Attributes

- `_CharT _M_delim`
- `_CharT _M_escape`
- `_String _M_string`

#### 5.495.1 Detailed Description

```
template<typename _String, typename _CharT>
struct std::__detail::__Quoted_string< _String, _CharT >
```

Struct for delimited strings.

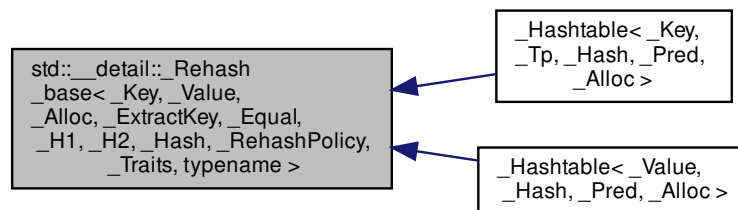
Definition at line 49 of file quoted\_string.h.

The documentation for this struct was generated from the following file:

- [quoted\\_string.h](#)

#### 5.496 std::\_\_detail::\_\_Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, typename > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_\_Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, typename >:



#### 5.496.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, typename = __detected_or_t<std::false_type, __has_load_factor, _RehashPolicy>>
struct std::__detail::__Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename >
```

Primary class template \_Rehash\_base.

Give hashtable the max\_load\_factor functions and reserve iff the rehash policy supports it.

Definition at line 1043 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

5.497 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::false_type >` Struct Template Reference

5.497.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::false_type >
```

Specialization when rehash policy doesn't provide load factor management.

Definition at line 1050 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

5.498 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::true_type >` Struct Template Reference

Public Types

- using `__hashtable` = `_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Public Member Functions

- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_z)
- void `reserve` (std::size\_t \_\_n)

5.498.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::true_type >
```

Specialization when rehash policy provide load factor management.

Definition at line 1061 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 5.499 `std::__detail::_Scanner<_CharT>` Class Template Reference

Inherits `std::__detail::_ScannerBase`.

### Public Types

- typedef const [std::ctype](#)<\_CharT> **\_CtypeT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef const \_CharT \* **\_IterT**
- typedef [std::basic\\_string](#)<\_CharT> **\_StringT**
- enum **\_TokenT** : unsigned {  
     **\_S\_token\_anychar**, **\_S\_token\_ord\_char**, **\_S\_token\_oct\_num**, **\_S\_token\_hex\_num**,  
     **\_S\_token\_backref**, **\_S\_token\_subexpr\_begin**, **\_S\_token\_subexpr\_no\_group\_begin**, **\_S\_token\_subexpr\_**↵  
     **\_lookahead\_begin**,  
     **\_S\_token\_subexpr\_end**, **\_S\_token\_bracket\_begin**, **\_S\_token\_bracket\_neg\_begin**, **\_S\_token\_bracket\_**↵  
     **end**,  
     **\_S\_token\_interval\_begin**, **\_S\_token\_interval\_end**, **\_S\_token\_quoted\_class**, **\_S\_token\_char\_class\_name**,  
     **\_S\_token\_collsymbol**, **\_S\_token\_equiv\_class\_name**, **\_S\_token\_opt**, **\_S\_token\_or**,  
     **\_S\_token\_closure0**, **\_S\_token\_closure1**, **\_S\_token\_line\_begin**, **\_S\_token\_line\_end**,  
     **\_S\_token\_word\_bound**, **\_S\_token\_comma**, **\_S\_token\_dup\_count**, **\_S\_token\_eof**,  
     **\_S\_token\_bracket\_dash**, **\_S\_token\_unknown** }

### Public Member Functions

- **\_Scanner** (\_IterT \_\_begin, \_IterT \_\_end, [\\_FlagT](#) \_\_flags, [std::locale](#) \_\_loc)
- void **\_M\_advance** ()
- [\\_TokenT](#) **\_M\_get\_token** () const
- const [\\_StringT](#) & **\_M\_get\_value** () const

### Protected Types

- enum **\_StateT** { **\_S\_state\_normal**, **\_S\_state\_in\_brace**, **\_S\_state\_in\_bracket** }

### Protected Member Functions

- const char \* **\_M\_find\_escape** (char \_\_c)
- bool **\_M\_is\_awk** () const
- bool **\_M\_is\_basic** () const
- bool **\_M\_is\_ecma** () const
- bool **\_M\_is\_extended** () const
- bool **\_M\_is\_grep** () const

## Protected Attributes

- `bool _M_at_bracket_start`
- `const std::pair< char, char > _M_awk_escape_tbl [11]`
- `const char * _M_basic_spec_char`
- `const std::pair< char, char > _M_ecma_escape_tbl [8]`
- `const char * _M_ecma_spec_char`
- `const std::pair< char, char > * _M_escape_tbl`
- `const char * _M_extended_spec_char`
- `_FlagT _M_flags`
- `const char * _M_spec_char`
- `_StateT _M_state`
- `_TokenT _M_token`
- `const std::pair< char, _TokenT > _M_token_tbl [9]`

## 5.499.1 Detailed Description

```
template<typename _CharT>
class std::__detail::_Scanner<_CharT>
```

Scans an input range for regex tokens.

The `_Scanner` class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

Definition at line 210 of file `regex_scanner.h`.

## 5.499.2 Member Enumeration Documentation

5.499.2.1 `_TokenT`

```
enum std::__detail::_ScannerBase::_TokenT : unsigned [inherited]
```

Token types returned from the scanner.

Definition at line 46 of file `regex_scanner.h`.

The documentation for this class was generated from the following files:

- [regex\\_scanner.h](#)
- [regex\\_scanner.tcc](#)



## 5.500 `std::__detail::_StateSeq<_TraitsT >` Class Template Reference

### Public Types

- `typedef _NFA<_TraitsT > _RegexT`

### Public Member Functions

- `_StateSeq` (`_RegexT &__nfa, _StateIdT __s`)
- `_StateSeq` (`_RegexT &__nfa, _StateIdT __s, _StateIdT __end`)
- `void _M_append` (`_StateIdT __id`)
- `void _M_append` (`const _StateSeq &__s`)
- `_StateSeq _M_clone` ()

### Public Attributes

- `_StateIdT _M_end`
- `_RegexT & _M_nfa`
- `_StateIdT _M_start`

#### 5.500.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_StateSeq<_TraitsT >
```

Describes a sequence of one or more `_State`, its current start and end(s). This structure contains fragments of an NFA during construction.

Definition at line 355 of file `regex_automaton.h`.

The documentation for this class was generated from the following files:

- [regex\\_automaton.h](#)
- [regex\\_automaton.tcc](#)

## 5.501 `std::__detector<_Default, _AlwaysVoid, _Op, _Args >` Struct Template Reference

### Public Types

- using `type` = `_Default`
- using `value_t` = `false_type`

## 5.501.1 Detailed Description

```
template<typename _Default, typename _AlwaysVoid, template< typename... > class _Op, typename... _Args>
struct std::__detector< _Default, _AlwaysVoid, _Op, _Args >
```

Implementation of the detection idiom (negative case).

Definition at line 2322 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.502 std::\_\_detector&lt; \_Default, \_\_void\_t&lt; \_Op&lt; \_Args... &gt; &gt;, \_Op, \_Args... &gt; Struct Template Reference

## Public Types

- using **type** = \_Op< \_Args... >
- using **value\_t** = [true\\_type](#)

## 5.502.1 Detailed Description

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
struct std::__detector< _Default, __void_t< _Op< _Args... > >, _Op, _Args... >
```

Implementation of the detection idiom (positive case).

Definition at line 2331 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.503 std::\_\_exception\_ptr::exception\_ptr Class Reference

## Public Member Functions

- **exception\_ptr** (const [exception\\_ptr](#) &) noexcept
- **exception\_ptr** (nullptr\_t) noexcept
- **exception\_ptr** ([exception\\_ptr](#) &&\_\_o) noexcept
- const class [std::type\\_info](#) \* **\_\_cxa\_exception\_type** () const noexcept \_\_attribute\_\_((\_\_pure\_\_))
- **operator bool** () const
- [exception\\_ptr](#) & **operator=** (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) & **operator=** ([exception\\_ptr](#) &&\_\_o) noexcept
- void **swap** ([exception\\_ptr](#) &) noexcept

## Friends

- `bool operator==(const exception\_ptr &, const exception\_ptr &) noexcept __attribute__((__pure__))`
- `exception\_ptr std::current_exception() noexcept`
- `template<typename _Ex >  
exception\_ptr std::make_exception_ptr(_Ex) noexcept`
- `void std::rethrow_exception(exception\_ptr)`

## 5.503.1 Detailed Description

An opaque pointer to an arbitrary exception.

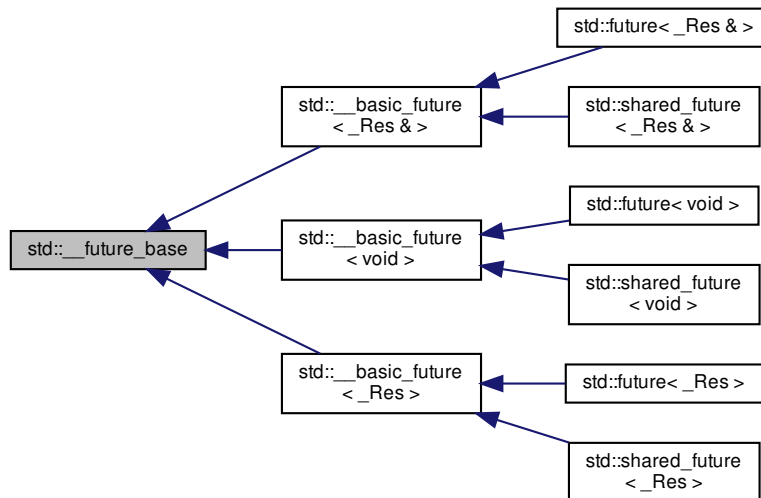
Definition at line 79 of file `exception_ptr.h`.

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

5.504 `std::__future_base` Struct Reference

Inheritance diagram for `std::__future_base`:



## Classes

- `struct \_Result`
- `struct \_Result<\_Res &>`
- `struct \_Result<void>`
- `struct \_Result\_alloc`
- `struct \_Result\_base`

## Public Types

- template<typename \_Res >  
using [\\_Ptr](#) = [unique\\_ptr](#)<\_Res, \_Result\_base::\_Deleter >
- using [\\_State\\_base](#) = \_State\_baseV2

## Static Public Member Functions

- template<typename \_Res , typename \_Allocator >  
static [\\_Ptr](#)< [\\_Result\\_alloc](#)< \_Res, \_Allocator > > [\\_S\\_allocate\\_result](#) (const \_Allocator &\_\_a)
- template<typename \_Res , typename \_Tp >  
static [\\_Ptr](#)< [\\_Result](#)< \_Res > > [\\_S\\_allocate\\_result](#) (const [std::allocator](#)< \_Tp > &\_\_a)
- template<typename \_BoundFn >  
static [std::shared\\_ptr](#)< \_State\_base > [\\_S\\_make\\_async\\_state](#) (\_BoundFn &&\_\_fn)
- template<typename \_BoundFn >  
static [std::shared\\_ptr](#)< \_State\_base > [\\_S\\_make\\_deferred\\_state](#) (\_BoundFn &&\_\_fn)
- template<typename \_Res\_ptr , typename \_BoundFn >  
static [\\_Task\\_setter](#)< \_Res\_ptr, \_BoundFn > [\\_S\\_task\\_setter](#) (\_Res\_ptr &\_\_ptr, \_BoundFn &\_\_call)

### 5.504.1 Detailed Description

Base class and enclosing scope.

Definition at line 198 of file future.

### 5.504.2 Member Typedef Documentation

#### 5.504.2.1 [\\_Ptr](#)

```
template<typename _Res >
using std::\_\_future\_base::\_Ptr = unique\_ptr<_Res, _Result_base::_Deleter>
```

A [unique\\_ptr](#) for result objects.

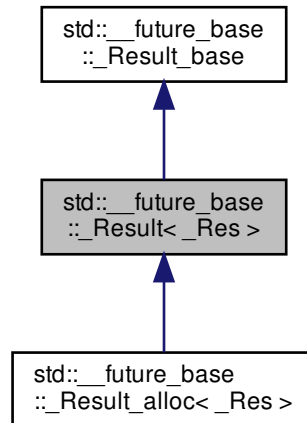
Definition at line 223 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

### 5.505 std::\_\_future\_base::\_Result< \_Res > Struct Template Reference

Inheritance diagram for std::\_\_future\_base::\_Result< \_Res >:



#### Public Types

- typedef `_Res` **result\_type**

#### Public Member Functions

- void **\_M\_set** (const `_Res` &\_\_res)
- void **\_M\_set** (`_Res` &&\_\_res)
- `_Res` & **\_M\_value** () noexcept

#### Public Attributes

- [exception\\_ptr](#) **\_M\_error**

#### 5.505.1 Detailed Description

```
template<typename _Res>
struct std::__future_base::_Result< _Res >
```

A result object that has storage for an object of type `_Res`.

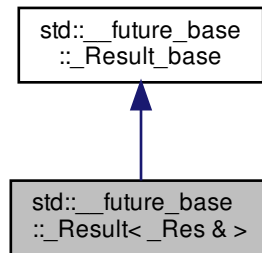
Definition at line 227 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.506 std::\_\_future\_base::\_\_Result&lt; \_Res &amp; &gt; Struct Template Reference

Inheritance diagram for std::\_\_future\_base::\_\_Result< \_Res & >:



## Public Types

- typedef `_Res &` **result\_type**

## Public Member Functions

- `_Res & _M_get ()` noexcept
- `void _M_set (_Res &__res)` noexcept

## Public Attributes

- [exception\\_ptr](#) **\_M\_error**

## 5.506.1 Detailed Description

```
template<typename _Res>
struct std::__future_base::__Result< _Res & >
```

Partial specialization for reference types.

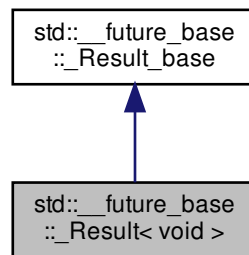
Definition at line 627 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

### 5.507 `std::__future_base::_Result< void >` Struct Template Reference

Inheritance diagram for `std::__future_base::_Result< void >`:



#### Public Types

- typedef void **result\_type**

#### Public Attributes

- [exception\\_ptr](#) **\_M\_error**

#### 5.507.1 Detailed Description

```
template<>
struct std::__future_base::_Result< void >
```

Explicit specialization for void.

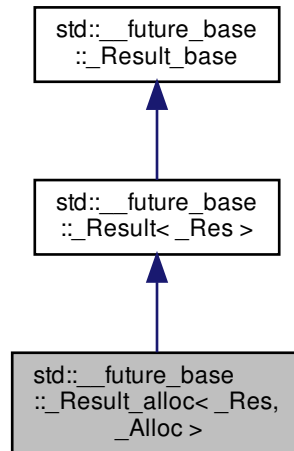
Definition at line 647 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.508 std::\_\_future\_base::\_\_Result\_alloc&lt; \_Res, \_Alloc &gt; Struct Template Reference

Inheritance diagram for std::\_\_future\_base::\_\_Result\_alloc< \_Res, \_Alloc >:



## Public Types

- using **\_\_allocator\_type** = \_\_alloc\_rebind< \_Alloc, [\\_Result\\_alloc](#) >
- typedef \_Res **result\_type**

## Public Member Functions

- **\_Result\_alloc** (const \_Alloc &\_\_a)
- void **\_M\_set** (const \_Res &\_\_res)
- void **\_M\_set** (\_Res &&\_\_res)
- \_Res & **\_M\_value** () noexcept

## Public Attributes

- [exception\\_ptr](#) **\_M\_error**

## 5.508.1 Detailed Description

```
template<typename _Res, typename _Alloc>
struct std::__future_base::__Result_alloc< _Res, _Alloc >
```

A result object that uses an allocator.

Definition at line 268 of file future.

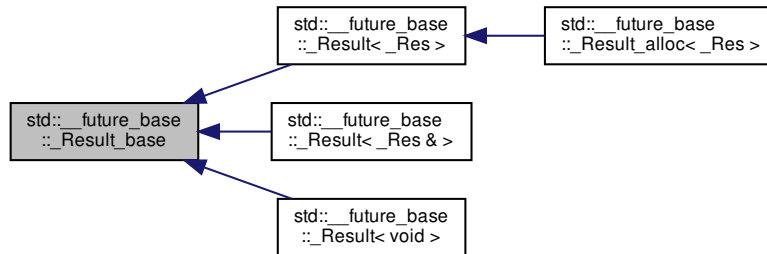
The documentation for this struct was generated from the following file:

- [future](#)



### 5.509 `std::__future_base::_Result_base` Struct Reference

Inheritance diagram for `std::__future_base::_Result_base`:



#### Public Member Functions

- `_Result_base` (const `_Result_base` &)=delete
- virtual void `_M_destroy` ()=0
- `_Result_base` & `operator=` (const `_Result_base` &)=delete

#### Public Attributes

- `exception_ptr _M_error`

#### 5.509.1 Detailed Description

Base class for results.

Definition at line 201 of file future.

The documentation for this struct was generated from the following file:

- `future`

### 5.510 `std::__is_location_invariant<_Tp>` Struct Template Reference

Inherits `type<_Tp>`.

Inherited by `std::__is_location_invariant<_Simple_type_wrapper<_Tp>>`.

## 5.510.1 Detailed Description

```
template<typename _Tp>
struct std::__is_location_invariant<_Tp>
```

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Trivially copyable types are location-invariant and users can specialize this trait for other types.

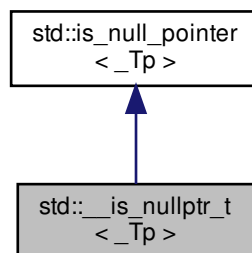
Definition at line 71 of file `std_function.h`.

The documentation for this struct was generated from the following file:

- [std\\_function.h](#)

5.511 `std::__is_nullptr_t<_Tp>` Struct Template Reference

Inheritance diagram for `std::__is_nullptr_t<_Tp>`:



## 5.511.1 Detailed Description

```
template<typename _Tp>
struct std::__is_nullptr_t<_Tp>
```

`__is_nullptr_t` (extension).

Definition at line 560 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.512 `std::__is_trivially_copy_assignable_impl<_Tp, bool>` Struct Template Reference

### 5.512.1 Detailed Description

```
template<typename _Tp, bool = __is_referenceable<_Tp>::value>
struct std::__is_trivially_copy_assignable_impl<_Tp, bool>
```

`is_trivially_copy_assignable`

Definition at line 1181 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.513 `std::__is_trivially_copy_constructible_impl<_Tp, bool>` Struct Template Reference

### 5.513.1 Detailed Description

```
template<typename _Tp, bool = __is_referenceable<_Tp>::value>
struct std::__is_trivially_copy_constructible_impl<_Tp, bool>
```

`is_trivially_copy_constructible`

Definition at line 1133 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.514 `std::__is_trivially_move_assignable_impl<_Tp, bool>` Struct Template Reference

### 5.514.1 Detailed Description

```
template<typename _Tp, bool = __is_referenceable<_Tp>::value>
struct std::__is_trivially_move_assignable_impl<_Tp, bool>
```

`is_trivially_move_assignable`

Definition at line 1202 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.515 std::\_\_is\_trivially\_move\_constructible\_impl&lt; \_Tp, bool &gt; Struct Template Reference

## 5.515.1 Detailed Description

```
template<typename _Tp, bool = __is_referenceable<_Tp>::value>
struct std::__is_trivially_move_constructible_impl< _Tp, bool >
```

is\_trivially\_move\_constructible

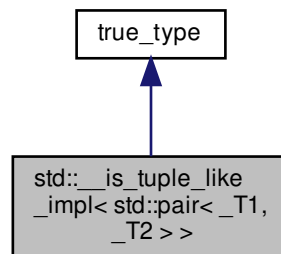
Definition at line 1154 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.516 std::\_\_is\_tuple\_like\_impl&lt; std::pair&lt; \_T1, \_T2 &gt; &gt; Struct Template Reference

Inheritance diagram for std::\_\_is\_tuple\_like\_impl< std::pair< \_T1, \_T2 > >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.516.1 Detailed Description

```
template<typename _T1, typename _T2>
struct std::__is_tuple_like_impl< std::pair< _T1, _T2 > >
```

Partial specialization for `std::pair`.

Definition at line 145 of file `utility`.

The documentation for this struct was generated from the following file:

- [utility](#)

#### 5.517 `std::__iterator_traits< _Iterator, typename >` Struct Template Reference

##### 5.517.1 Detailed Description

```
template<typename _Iterator, typename = __void_t<>>
struct std::__iterator_traits< _Iterator, typename >
```

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the `Iterator` argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

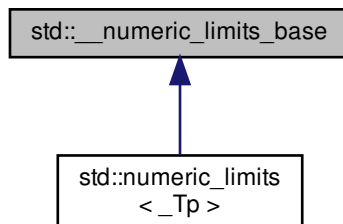
Definition at line 144 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 5.518 `std::__numeric_limits_base` Struct Reference

Inheritance diagram for `std::__numeric_limits_base`:



### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

#### 5.518.1 Detailed Description

Part of std::numeric\_limits.

The `static const` members are usable as integral constant expressions.

#### Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the std::numeric\_limits class.

Definition at line 202 of file limits.

#### 5.518.2 Member Data Documentation

#### 5.518.2.1 digits

```
constexpr int std::__numeric_limits_base::digits [static]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 211 of file `limits`.

#### 5.518.2.2 digits10

```
constexpr int std::__numeric_limits_base::digits10 [static]
```

The number of base 10 digits that can be represented without change.

Definition at line 214 of file `limits`.

#### 5.518.2.3 has\_denorm

```
constexpr float_denorm_style std::__numeric_limits_base::has_denorm [static]
```

See `std::float_denorm_style` for more information.

Definition at line 266 of file `limits`.

#### 5.518.2.4 has\_denorm\_loss

```
constexpr bool std::__numeric_limits_base::has_denorm_loss [static]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 270 of file `limits`.

#### 5.518.2.5 has\_infinity

```
constexpr bool std::__numeric_limits_base::has_infinity [static]
```

True if the type has a representation for positive infinity.

Definition at line 255 of file `limits`.

#### 5.518.2.6 has\_quiet\_NaN

```
constexpr bool std::__numeric_limits_base::has_quiet_NaN [static]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 259 of file limits.

#### 5.518.2.7 has\_signaling\_NaN

```
constexpr bool std::__numeric_limits_base::has_signaling_NaN [static]
```

True if the type has a representation for a signaling Not a Number.

Definition at line 263 of file limits.

#### 5.518.2.8 is\_bounded

```
constexpr bool std::__numeric_limits_base::is_bounded [static]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 279 of file limits.

#### 5.518.2.9 is\_exact

```
constexpr bool std::__numeric_limits_base::is_exact [static]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 231 of file limits.

#### 5.518.2.10 is\_iec559

```
constexpr bool std::__numeric_limits_base::is_iec559 [static]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 274 of file limits.



#### 5.518.2.11 `is_integer`

```
constexpr bool std::__numeric_limits_base::is_integer [static]
```

True if the type is integer.

Definition at line 226 of file limits.

#### 5.518.2.12 `is_modulo`

```
constexpr bool std::__numeric_limits_base::is_modulo [static]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max()]`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 288 of file limits.

#### 5.518.2.13 `is_signed`

```
constexpr bool std::__numeric_limits_base::is_signed [static]
```

True if the type is signed.

Definition at line 223 of file limits.

#### 5.518.2.14 `is_specialized`

```
constexpr bool std::__numeric_limits_base::is_specialized [static]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 206 of file limits.

#### 5.518.2.15 `max_digits10`

```
constexpr int std::__numeric_limits_base::max_digits10 [static]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 219 of file limits.

#### 5.518.2.16 max\_exponent

```
constexpr int std::__numeric_limits_base::max_exponent [static]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 248 of file `limits`.

#### 5.518.2.17 max\_exponent10

```
constexpr int std::__numeric_limits_base::max_exponent10 [static]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 252 of file `limits`.

#### 5.518.2.18 min\_exponent

```
constexpr int std::__numeric_limits_base::min_exponent [static]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 239 of file `limits`.

#### 5.518.2.19 min\_exponent10

```
constexpr int std::__numeric_limits_base::min_exponent10 [static]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 243 of file `limits`.

#### 5.518.2.20 radix

```
constexpr int std::__numeric_limits_base::radix [static]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 235 of file `limits`.

#### 5.518.2.21 round\_style

```
constexpr float_round_style std::__numeric_limits_base::round_style [static]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 299 of file `limits`.

#### 5.518.2.22 tinyness\_before

```
constexpr bool std::__numeric_limits_base::tinyness_before [static]
```

True if tininess is detected before rounding. (see IEC 559)

Definition at line 294 of file `limits`.

#### 5.518.2.23 traps

```
constexpr bool std::__numeric_limits_base::traps [static]
```

True if trapping is implemented for this type.

Definition at line 291 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.519 std::\_\_parallel::\_CRandNumber<\_MustBeInt > Struct Template Reference

#### Public Member Functions

- `int operator() (int __limit)`

#### 5.519.1 Detailed Description

```
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber<_MustBeInt >
```

Functor wrapper for `std::rand()`.

Definition at line 1529 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

## 5.520 std::\_\_profile::bitset&lt; \_Nb &gt; Class Template Reference

Inherits `bitset< _Nb >`.

## Public Member Functions

- constexpr **bitset** (unsigned long long \_\_val) noexcept
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n=([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::npos))
- template<class \_CharT, class \_Traits, class \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- **bitset** (const [\\_Base](#) &\_\_x)
- template<typename \_CharT >  
**bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- [\\_Base](#) & **M\_base** () noexcept
- const [\\_Base](#) & **M\_base** () const noexcept
- [bitset](#)< \_Nb > & **flip** () noexcept
- [bitset](#)< \_Nb > & **flip** (size\_t \_\_pos)
- bool **operator!=** (const [bitset](#)< \_Nb > &\_\_rhs) const noexcept
- [bitset](#)< \_Nb > & **operator&=** (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > **operator<<** (size\_t \_\_pos) const noexcept
- [bitset](#)< \_Nb > & **operator<<=** (size\_t \_\_pos) noexcept
- bool **operator==** (const [bitset](#)< \_Nb > &\_\_rhs) const noexcept
- [bitset](#)< \_Nb > **operator>>** (size\_t \_\_pos) const noexcept
- [bitset](#)< \_Nb > & **operator>>=** (size\_t \_\_pos) noexcept
- [bitset](#)< \_Nb > & **operator^=** (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > & **operator|=** (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > **operator~** () const noexcept
- [bitset](#)< \_Nb > & **reset** () noexcept
- [bitset](#)< \_Nb > & **reset** (size\_t \_\_pos)
- [bitset](#)< \_Nb > & **set** () noexcept
- [bitset](#)< \_Nb > & **set** (size\_t \_\_pos, bool \_\_val=true)

## 5.520.1 Detailed Description

```
template<size_t _Nb>
class std::__profile::bitset< _Nb >
```

Class `std::bitset` wrapper with performance instrumentation, none at the moment.

Definition at line 41 of file `profile/bitset`.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

## 5.521 `std::__profile::deque<_Tp, _Allocator>` Class Template Reference

Inherits `deque<_Tp, _Allocator>`.

### Public Types

- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- `deque` (const `deque` &)=default
- `deque` (`deque` &&)=default
- `deque` (const `deque` &\_\_d, const `_Allocator` &\_\_a)
- `deque` (`deque` &&\_\_d, const `_Allocator` &\_\_a)
- `deque` (`initializer_list`< `value_type` > \_\_l, const `_Allocator` &\_\_a=\_Allocator())
- `deque` (const `_Allocator` &\_\_a)
- `deque` (size\_type \_\_n, const `_Allocator` &\_\_a=\_Allocator())
- `deque` (size\_type \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=\_Allocator())
- `template<typename _InputIterator, typename = std::::RequireInputIter<_InputIterator>>`  
`deque` (\_InputIterator \_\_first, \_InputIterator \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- `deque` (const `Base` &\_\_x)
- `_Base` & `_M_base` () noexcept
- const `_Base` & `_M_base` () const noexcept
- `deque` & `operator=` (const `deque` &)=default
- `deque` & `operator=` (`deque` &&)=default
- `deque` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- void `swap` (`deque` &\_\_x) noexcept(*/\*conditional \*/*)

### 5.521.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__profile::deque<_Tp, _Allocator>
```

Class `std::deque` wrapper with performance instrumentation.

Definition at line 40 of file `profile/deque`.

The documentation for this class was generated from the following file:

- `profile/deque`

## 5.522 `std::__profile::forward_list<_Tp, _Alloc>` Class Template Reference

Inherits `forward_list<_Tp, _Alloc>`.

## Public Types

- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::size\_type **size\_type**

## Public Member Functions

- **forward\_list** (const \_Alloc &\_\_al) noexcept
- **forward\_list** (const [forward\\_list](#) &\_\_list, const \_Alloc &\_\_al)
- **forward\_list** ([forward\\_list](#) &&\_\_list, const \_Alloc &\_\_al)
- **forward\_list** (size\_type \_\_n, const \_Alloc &\_\_al=\_Alloc())
- **forward\_list** (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al=\_Alloc())
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
**forward\_list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Alloc &\_\_al=\_Alloc())
- **forward\_list** (const [forward\\_list](#) &)=default
- **forward\_list** ([forward\\_list](#) &&)=default
- **forward\_list** (std::initializer\_list< \_Tp > \_\_il, const \_Alloc &\_\_al=\_Alloc())
- [\\_Base](#) & **\_M\_base** () noexcept
- const [\\_Base](#) & **\_M\_base** () const noexcept
- void **merge** ([forward\\_list](#) &&\_\_list)
- void **merge** ([forward\\_list](#) &\_\_list)
- template<typename \_Comp >  
void **merge** ([forward\\_list](#) &&\_\_list, \_Comp \_\_comp)
- template<typename \_Comp >  
void **merge** ([forward\\_list](#) &\_\_list, \_Comp \_\_comp)
- [forward\\_list](#) & **operator=** (const [forward\\_list](#) &)=default
- [forward\\_list](#) & **operator=** ([forward\\_list](#) &&)=default
- [forward\\_list](#) & **operator=** (std::initializer\_list< \_Tp > \_\_il)
- void **splice\_after** (const\_iterator \_\_pos, [forward\\_list](#) &&\_\_fl)
- void **splice\_after** (const\_iterator \_\_pos, [forward\\_list](#) &\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, [forward\\_list](#) &&\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, [forward\\_list](#) &\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, [forward\\_list](#) &&\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **splice\_after** (const\_iterator \_\_pos, [forward\\_list](#) &\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **swap** ([forward\\_list](#) &\_\_fl) noexcept(noexcept(declval< [\\_Base](#) & >().swap(\_\_fl)))

## 5.522.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__profile::forward_list< _Tp, _Alloc >
```

Class std::forward\_list wrapper with performance instrumentation.

Definition at line 44 of file profile/forward\_list.

The documentation for this class was generated from the following file:

- [profile/forward\\_list](#)

### 5.523 `std::__profile::list<_Tp, _Allocator>` Class Template Reference

Inherits `list<_Tp, _Allocator>`, and `std::__profile::_List_profile<_List>`.

#### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__iterator_tracker<typename _Base::const_iterator, list>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__iterator_tracker<typename _Base::iterator, list>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- **list** (const `list` &)=default
- **list** (`list` &&)=default
- **list** (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a=`allocator_type`())
- **list** (const `list` &\_\_x, const `allocator_type` &\_\_a)
- **list** (`list` &&\_\_x, const `allocator_type` &\_\_a)
- **list** (const `_Allocator` &\_\_a) noexcept
- **list** (`size_type` \_\_n, const `allocator_type` &\_\_a=`allocator_type`())
- **list** (`size_type` \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=`_Allocator`())
- template<typename `_InputIterator` , typename = `std::_RequireInputIter<_InputIterator>`>>  
**list** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=`_Allocator`())
- **list** (const `_Base` &\_\_x)
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_profile\_construct** () noexcept
- void **\_M\_profile\_destruct** () noexcept
- void **\_M\_profile\_iterate** (int \_\_rewind=0) const
- void **\_M\_swap** (`_List_profile` &\_\_other)
- reference **back** () noexcept
- const\_reference **back** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- `const_reverse_iterator` **crbegin** () const noexcept
- `const_reverse_iterator` **crend** () const noexcept
- template<typename... `_Args`>  
iterator **emplace** (const\_iterator \_\_position, `_Args` &&... \_\_args)

- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (const\_iterator \_\_pos) noexcept
- iterator **erase** (const\_iterator \_\_pos, const\_iterator \_\_last) noexcept
- iterator **insert** (const\_iterator \_\_pos, const \_Tp &\_\_x)
- iterator **insert** (const\_iterator \_\_pos, \_Tp &&\_\_x)
- iterator **insert** (const\_iterator \_\_pos, [initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_x)
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>  
iterator **insert** (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **merge** ([list](#) &&\_\_x)
- void **merge** ([list](#) &\_\_x)
- template<class \_Compare >  
void **merge** ([list](#) &&\_\_x, \_Compare \_\_comp)
- template<typename \_Compare >  
void **merge** ([list](#) &\_\_x, \_Compare \_\_comp)
- [list](#) & **operator=** (const [list](#) &)=default
- [list](#) & **operator=** ([list](#) &&)=default
- [list](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **pop\_back** () noexcept
- void **pop\_front** () noexcept
- void **push\_front** (const value\_type &\_\_x)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- void **remove** (const \_Tp &\_\_value)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **splice** (const\_iterator \_\_pos, [list](#) &&\_\_x) noexcept
- void **splice** (const\_iterator \_\_pos, [list](#) &\_\_x) noexcept
- void **splice** (const\_iterator \_\_pos, [list](#) &\_\_x, const\_iterator \_\_i)
- void **splice** (const\_iterator \_\_pos, [list](#) &&\_\_x, const\_iterator \_\_i) noexcept
- void **splice** (const\_iterator \_\_pos, [list](#) &&\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void **splice** (const\_iterator \_\_pos, [list](#) &\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void **swap** ([list](#) &\_\_x) noexcept([/\\*conditional \\*/](#))
- void **unique** ()
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_binary\_pred)

#### Public Attributes

- \_\_gnu\_profile::\_\_list2slist\_info \* **\_M\_list2slist\_info**
- \_\_gnu\_profile::\_\_list2vector\_info \* **\_M\_list2vector\_info**



### 5.523.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__profile::list< _Tp, _Allocator >
```

List wrapper with performance instrumentation.

Definition at line 106 of file profile/list.

The documentation for this class was generated from the following file:

- [profile/list](#)

### 5.524 std::\_\_profile::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference

Inherits map< \_Key, \_Tp, \_Compare, \_Allocator >, and std::\_\_profile::\_\_Ordered\_profile< \_Cont >.

#### Public Types

- typedef \_\_iterator\_tracker< \_Base\_const\_iterator, [map](#) > **const\_iterator**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_\_iterator\_tracker< \_Base\_iterator, [map](#) > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

#### Public Member Functions

- **map** (const [map](#) &)=default
- **map** ([map](#) &&)=default
- **map** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>
 **map** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **map** (const [\\_Base](#) &\_\_x)
- **map** ([initializer\\_list](#)< value\_type > \_\_l, const \_Compare &\_\_c=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **map** (const \_Allocator &\_\_a)
- **map** (const [map](#) &\_\_x, const \_Allocator &\_\_a)
- **map** ([map](#) &&\_\_x, const \_Allocator &\_\_a) noexcept(noexcept([\\_Base](#)(std::move(\_\_x), \_\_a)))
- **map** ([initializer\\_list](#)< value\_type > \_\_l, const \_Allocator &\_\_a)

- `template<typename _InputIterator >`  
`map` ( `_InputIterator __first`, `_InputIterator __last`, `const _Allocator &__a`)
- `_Base & _M_base` () noexcept
- `const _Base & _M_base` () const noexcept
- `void _M_profile_iterate` (int `__rewind=0`) const
- `mapped_type & at` (const `key_type &__k`)
- `const mapped_type & at` (const `key_type &__k`) const
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `size_type count` (const `key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`size_type count` (const `_Kt &__x`) const
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- `template<typename... _Args>`  
`std::pair< iterator, bool > emplace` ( `_Args &&... __args`)
- `template<typename... _Args>`  
`iterator emplace_hint` (const `iterator __pos`, `_Args &&... __args`)
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `std::pair< iterator, iterator > equal_range` (const `key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< iterator, iterator > equal_range` (const `_Kt &__x`)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< const_iterator, const_iterator > equal_range` (const `_Kt &__x`) const
- `iterator erase` (const `iterator __pos`)
- `iterator erase` (`iterator __pos`)
- `size_type erase` (const `key_type &__x`)
- `iterator erase` (const `iterator __first`, const `iterator __last`)
- `iterator find` (const `key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator find` (const `_Kt &__x`)
- `const_iterator find` (const `key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator find` (const `_Kt &__x`) const
- `std::pair< iterator, bool > insert` (const `value_type &__x`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`std::pair< iterator, bool > insert` ( `_Pair &&__x`)
- `void insert` (`std::initializer_list< value_type > __list`)
- `iterator insert` (const `iterator __pos`, const `value_type &__x`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert` (const `iterator __pos`, `_Pair &&__x`)
- `template<typename _InputIterator >`  
`void insert` ( `_InputIterator __first`, `_InputIterator __last`)
- `iterator lower_bound` (const `key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator lower_bound` (const `_Kt &__x`)

- `const_iterator` **lower\_bound** (`const key_type &__x`) `const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **lower\_bound** (`const _Kt &__x`) `const`
- `map & operator=` (`const map &`)=`default`
- `map & operator=` (`map &&`)=`default`
- `map & operator=` (`initializer_list< value_type > __l`)
- `mapped_type & operator[]` (`const key_type &__k`)
- `mapped_type & operator[]` (`key_type &&__k`)
- `reverse_iterator` **rbegin** () `noexcept`
- `const_reverse_iterator` **rbegin** () `const noexcept`
- `reverse_iterator` **rend** () `noexcept`
- `const_reverse_iterator` **rend** () `const noexcept`
- `void swap` (`map &__x`) `noexcept`(/\*`conditional` \*/)
- `iterator` **upper\_bound** (`const key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **upper\_bound** (`const _Kt &__x`)
- `const_iterator` **upper\_bound** (`const key_type &__x`) `const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **upper\_bound** (`const _Kt &__x`) `const`

#### Protected Member Functions

- `void` **\_M\_profile\_construct** () `noexcept`
- `void` **\_M\_profile\_destruct** () `noexcept`
- `void` **\_M\_swap** (`_Ordered_profile &__other`)

#### Protected Attributes

- `__gnu_profile::__map2umap_info` \* **\_M\_map2umap\_info**

#### Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool` **operator<** (`const map< _K1, _T1, _C1, _A1 > &`, `const map< _K1, _T1, _C1, _A1 > &`)
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool` **operator==** (`const map< _K1, _T1, _C1, _A1 > &`, `const map< _K1, _T1, _C1, _A1 > &`)

#### 5.524.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class `std::map` wrapper with performance instrumentation.

Definition at line 41 of file `profile/map.h`.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

## 5.525 std::\_\_profile::multimap&lt; \_Key, \_Tp, \_Compare, \_Allocator &gt; Class Template Reference

Inherits `multimap< _Key, _Tp, _Compare, _Allocator >`, and `std::__profile::_Ordered_profile< _Cont >`.

## Public Types

- typedef `__iterator_tracker< _Base_const_iterator, multimap >` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__iterator_tracker< _Base_iterator, multimap >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **multimap** (const [multimap](#) &)=default
- **multimap** ([multimap](#) &&)=default
- **multimap** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
**multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Compare &\_\_c=\_Compare(), const \_Allocator &\_\_a=\_↵  
Allocator())
- **multimap** (const \_Allocator &\_\_a)
- **multimap** (const [multimap](#) &\_\_x, const \_Allocator &\_\_a)
- **multimap** ([multimap](#) && \_\_x, const \_Allocator &\_\_a) noexcept(noexcept([\\_Base](#)(std::move(\_\_x), \_\_a))
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Allocator &\_\_a)
- template<typename \_InputIterator >  
**multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a)
- **multimap** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & **\_M\_base** () noexcept
- const [\\_Base](#) & **\_M\_base** () const noexcept
- void **\_M\_profile\_iterate** (int \_\_rewind=0) const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- size\_type **count** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
size\_type **count** (const \_Kt &\_\_x) const
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept

- `template<typename... _Args>`  
iterator **emplace** (`_Args &&... __args`)
- `template<typename... _Args>`  
iterator **emplace\_hint** (`const_iterator __pos, _Args &&... __args`)
- iterator **end** () noexcept
- `const_iterator end` () const noexcept
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< iterator, iterator >` **equal\_range** (`const _Kt &__x`)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< const_iterator, const_iterator >` **equal\_range** (`const _Kt &__x`) const
- iterator **erase** (`const_iterator __pos`)
- iterator **erase** (`iterator __pos`)
- `size_type erase` (`const key_type &__x`)
- iterator **erase** (`const_iterator __first, const_iterator __last`)
- iterator **find** (`const key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
iterator **find** (`const _Kt &__x`)
- `const_iterator find` (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator find` (`const _Kt &__x`) const
- iterator **insert** (`const value_type &__x`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
iterator **insert** (`_Pair &&__x`)
- void **insert** (`std::initializer_list< value_type > __list`)
- iterator **insert** (`const_iterator __pos, const value_type &__x`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
iterator **insert** (`const_iterator __pos, _Pair &&__x`)
- `template<typename _InputIterator >`  
void **insert** (`_InputIterator __first, _InputIterator __last`)
- iterator **lower\_bound** (`const key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
iterator **lower\_bound** (`const _Kt &__x`)
- `const_iterator lower_bound` (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator lower_bound` (`const _Kt &__x`) const
- `multimap & operator=` (`const multimap &`)=default
- `multimap & operator=` (`multimap &&`)=default
- `multimap & operator=` (`initializer_list< value_type > __l`)
- `reverse_iterator rbegin` () noexcept
- `const_reverse_iterator rbegin` () const noexcept
- `reverse_iterator rend` () noexcept
- `const_reverse_iterator rend` () const noexcept
- void **swap** (`multimap &__x`) noexcept(`/*conditional */`)
- iterator **upper\_bound** (`const key_type &__x`)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
iterator **upper\_bound** (`const _Kt &__x`)
- `const_iterator upper_bound` (`const key_type &__x`) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator upper_bound` (`const _Kt &__x`) const

## Protected Member Functions

- void **\_M\_profile\_construct** () noexcept
- void **\_M\_profile\_destruct** () noexcept
- void **\_M\_swap** (\_Ordered\_profile &\_other)

## Protected Attributes

- [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#) \* **\_M\_map2umap\_info**

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**< (const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &, const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**== (const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &, const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &)

## 5.525.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>
class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class std::multimap wrapper with performance instrumentation.

Definition at line 42 of file profile/multimap.h.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

## 5.526 std::\_\_profile::multiset&lt; \_Key, \_Compare, \_Allocator &gt; Class Template Reference

Inherits multiset< \_Key, \_Compare, \_Allocator >, and std::\_\_profile::\_\_Ordered\_profile< \_Cont >.

## Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_\_iterator\_tracker< \_Base\_const\_iterator, [multiset](#) > **const\_iterator**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_\_iterator\_tracker< \_Base\_iterator, [multiset](#) > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

## Public Member Functions

- **multiset** (const [multiset](#) &)=default
- **multiset** ([multiset](#) &&)=default
- **multiset** (const [\\_Compare](#) &\_\_comp, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- template<typename [\\_InputIterator](#) , typename = std::RequireInputIter<[\\_InputIterator](#)>>  
**multiset** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- **multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **multiset** (const [allocator\\_type](#) &\_\_a)
- **multiset** (const [multiset](#) &\_\_x, const [allocator\\_type](#) &\_\_a)
- **multiset** ([multiset](#) &&\_\_x, const [allocator\\_type](#) &\_\_a) noexcept(noexcept([\\_Base](#)(std::move(\_\_x), \_\_a)))
- **multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- template<typename [\\_InputIterator](#) >  
**multiset** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [allocator\\_type](#) &\_\_a)
- **multiset** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_profile\\_iterate](#) (int \_\_rewind=0) const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- [size\\_type](#) **count** (const [key\\_type](#) &\_\_x) const
- template<typename [\\_Kt](#) , typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[size\\_type](#) **count** (const [\\_Kt](#) &\_\_x) const
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- template<typename... [\\_Args](#)>  
iterator **emplace** ([\\_Args](#) &&... \_\_args)
- template<typename... [\\_Args](#)>  
iterator **emplace\_hint** (const\_iterator \_\_pos, [\\_Args](#) &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- [std::pair](#)< iterator, iterator > **equal\_range** (const [key\\_type](#) &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const [key\\_type](#) &\_\_x) const
- template<typename [\\_Kt](#) , typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[std::pair](#)< iterator, iterator > **equal\_range** (const [\\_Kt](#) &\_\_x)
- template<typename [\\_Kt](#) , typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const [\\_Kt](#) &\_\_x) const
- iterator **erase** (const\_iterator \_\_pos)
- [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const [key\\_type](#) &\_\_x)
- const\_iterator **find** (const [key\\_type](#) &\_\_x) const
- template<typename [\\_Kt](#) , typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
iterator **find** (const [\\_Kt](#) &\_\_x)
- template<typename [\\_Kt](#) , typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
const\_iterator **find** (const [\\_Kt](#) &\_\_x) const

- iterator **insert** (const value\_type &\_\_x)
- iterator **insert** (value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_pos, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_pos, value\_type &&\_\_x)
- template<typename \_InputIterator, typename = std::\_\_RequireInputIter<\_InputIterator>>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (initializer\_list< value\_type > \_\_l)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
iterator **lower\_bound** (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
const\_iterator **lower\_bound** (const \_Kt &\_\_x) const
- multiset & **operator=** (const multiset &)=default
- multiset & **operator=** (multiset &&)=default
- multiset & **operator=** (initializer\_list< value\_type > \_\_l)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **swap** (multiset &\_\_x) noexcept(/\*conditional \*/)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
iterator **upper\_bound** (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
const\_iterator **upper\_bound** (const \_Kt &\_\_x) const

#### Protected Member Functions

- void **\_M\_profile\_construct** () noexcept
- void **\_M\_profile\_destruct** () noexcept
- void **\_M\_swap** (\_Ordered\_profile &\_\_other)

#### Protected Attributes

- [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#) \* **\_M\_map2umap\_info**

#### Friends

- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator<** (const multiset< \_K1, \_C1, \_A1 > &, const multiset< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator==** (const multiset< \_K1, \_C1, \_A1 > &, const multiset< \_K1, \_C1, \_A1 > &)



### 5.526.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__profile::multiset< _Key, _Compare, _Allocator >
```

Class std::multiset wrapper with performance instrumentation.

Definition at line 42 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

### 5.527 std::\_\_profile::set< \_Key, \_Compare, \_Allocator > Class Template Reference

Inherits set< \_Key, \_Compare, \_Allocator >, and std::\_\_profile::\_Ordered\_profile< \_Cont >.

#### Public Types

- typedef \_\_iterator\_tracker< \_Base\_const\_iterator, [set](#) > **const\_iterator**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_\_iterator\_tracker< \_Base\_iterator, [set](#) > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

#### Public Member Functions

- **set** (const [set](#) &)=default
- **set** ([set](#) &&)=default
- **set** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
**set** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **set** ([initializer\\_list](#)< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **set** (const \_Allocator &\_\_a)
- **set** (const [set](#) &\_\_x, const \_Allocator &\_\_a)
- **set** ([set](#) &&\_\_x, const \_Allocator &\_\_a) noexcept(noexcept([\\_Base](#)(std::move(\_\_x), \_\_a)))
- **set** ([initializer\\_list](#)< value\_type > \_\_l, const \_Allocator &\_\_a)

- template<typename \_InputIterator >  
set (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a)
- set (const \_Base &\_\_x)
- \_Base & \_M\_base () noexcept
- const \_Base & \_M\_base () const noexcept
- void \_M\_profile\_iterate (int \_\_rewind=0) const
- iterator begin () noexcept
- const\_iterator begin () const noexcept
- const\_iterator cbegin () const noexcept
- const\_iterator cend () const noexcept
- void clear () noexcept
- size\_type count (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
size\_type count (const \_Kt &\_\_x) const
- const\_reverse\_iterator crbegin () const noexcept
- const\_reverse\_iterator crend () const noexcept
- template<typename... \_Args>  
std::pair< iterator, bool > emplace (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator emplace\_hint (const\_iterator \_\_pos, \_Args &&... \_\_args)
- iterator end () noexcept
- const\_iterator end () const noexcept
- std::pair< iterator, iterator > equal\_range (const key\_type &\_\_x)
- std::pair< const\_iterator, const\_iterator > equal\_range (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
std::pair< iterator, iterator > equal\_range (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
std::pair< const\_iterator, const\_iterator > equal\_range (const \_Kt &\_\_x) const
- iterator erase (const\_iterator \_\_pos)
- size\_type erase (const key\_type &\_\_x)
- iterator erase (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator find (const key\_type &\_\_x)
- const\_iterator find (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
iterator find (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
const\_iterator find (const \_Kt &\_\_x) const
- std::pair< iterator, bool > insert (const value\_type &\_\_x)
- std::pair< iterator, bool > insert (value\_type &&\_\_x)
- iterator insert (const\_iterator \_\_pos, const value\_type &\_\_x)
- iterator insert (const\_iterator \_\_pos, value\_type &&\_\_x)
- template<typename \_InputIterator >  
void insert (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void insert (initializer\_list< value\_type > \_\_l)
- iterator lower\_bound (const key\_type &\_\_x)
- const\_iterator lower\_bound (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
iterator lower\_bound (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
const\_iterator lower\_bound (const \_Kt &\_\_x) const
- set & operator= (const set &)=default

- [set](#) & **operator=** ([set](#) &&)=default
- [set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **swap** ([set](#) &\_\_x) noexcept(*/\*conditional \*/*)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
iterator **upper\_bound** (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
const\_iterator **upper\_bound** (const \_Kt &\_\_x) const

#### Protected Member Functions

- void **\_M\_profile\_construct** () noexcept
- void **\_M\_profile\_destruct** () noexcept
- void **\_M\_swap** (\_Ordered\_profile &\_\_other)

#### Protected Attributes

- [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#) \* **\_M\_map2umap\_info**

#### Friends

- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator<** (const [set](#)< \_K1, \_C1, \_A1 > &, const [set](#)< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator==** (const [set](#)< \_K1, \_C1, \_A1 > &, const [set](#)< \_K1, \_C1, \_A1 > &)

#### 5.527.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__profile::set< _Key, _Compare, _Allocator >
```

Class std::set wrapper with performance instrumentation.

Definition at line 42 of file profile/set.h.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

## 5.528 std::\_\_profile::unordered\_map&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inherits unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, and std::\_\_profile::Unordered\_profile< \_Unordered↵  
Cont, \_Unique\_keys >.

## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- typedef \_Base::iterator **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_Base::mapped\_type **mapped\_type**
- typedef \_Base::reference **reference**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const [unordered\\_map](#) &)=default
- **unordered\_map** (const [\\_Base](#) &\_\_x)
- **unordered\_map** ([unordered\\_map](#) &&)=default
- **unordered\_map** (const allocator\_type &\_\_a)
- **unordered\_map** (const [unordered\\_map](#) &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** ([unordered\\_map](#) &&\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key↵  
\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- void **clear** () noexcept
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > **emplace** (\_Args &&... \_\_args)

- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __it, _Args &&... __args)`
- `void insert (std::initializer_list< value_type > __l)`
- `std::pair< iterator, bool > insert (const value_type &__obj)`
- `iterator insert (const_iterator __iter, const value_type &__v)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`std::pair< iterator, bool > insert (_Pair &&__obj)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert (const_iterator __iter, _Pair &&__v)`
- `template<typename _InputIter >`  
`void insert (_InputIter __first, _InputIter __last)`
- `unordered_map & operator= (const unordered_map &)=default`
- `unordered_map & operator= (unordered_map &&)=default`
- `unordered_map & operator= (initializer_list< value_type > __l)`
- `mapped_type & operator[] (const _Key &__k)`
- `mapped_type & operator[] (_Key &&__k)`
- `void rehash (size_type __n)`
- `void swap (unordered_map &__x) noexcept(noexcept(__x._M_base().swap(__x)))`

#### Protected Member Functions

- `void _M_profile_construct () noexcept`
- `void _M_profile_destruct () noexcept`
- `void _M_profile_resize (std::size_t __old_size)`
- `void _M_swap (_Unordered_profile &__other) noexcept`

#### Protected Attributes

- `__gnu_profile::__hashfunc_info * _M_hashfunc_info`
- `__gnu_profile::__container_size_info * _M_size_info`

#### 5.528.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<std::pair<const _Key, _Tp>>>>
class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_map` wrapper with performance instrumentation.

Definition at line 51 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- `profile/unordered_map`

## 5.529 std::\_\_profile::unordered\_multimap&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inherits `unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, and `std::__profile::Unordered_profile< _Key, _Tp, _Hash, _Pred, _Alloc, _Unique_keys >`.

## Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const [unordered\\_multimap](#) &)=default
- **unordered\_multimap** (const [\\_Base](#) &\_\_x)
- **unordered\_multimap** ([unordered\\_multimap](#) &&)=default
- **unordered\_multimap** (const allocator\_type &\_\_a)
- **unordered\_multimap** (const [unordered\\_multimap](#) &\_\_ummap, const allocator\_type &\_\_a)
- **unordered\_multimap** ([unordered\\_multimap](#) &&\_\_ummap, const allocator\_type &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- `template<typename _InputIterator >`  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- `template<typename _InputIterator >`  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- void **clear** () noexcept
- `template<typename... _Args>`  
iterator **emplace** (\_Args &&... \_\_args)
- `template<typename... _Args>`  
iterator **emplace\_hint** (const\_iterator \_\_it, \_Args &&... \_\_args)

- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (\_Pair &&\_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_iter, \_Pair &&\_\_v)
- template<typename \_InputIter >  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- [unordered\\_multimap](#) & **operator=** (const [unordered\\_multimap](#) &)=default
- [unordered\\_multimap](#) & **operator=** ([unordered\\_multimap](#) &&)=default
- [unordered\\_multimap](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_multimap](#) &\_\_x) noexcept(noexcept(\_\_x.\_M\_base().swap(\_\_x)))

#### Protected Member Functions

- void **\_M\_profile\_construct** () noexcept
- void **\_M\_profile\_destruct** () noexcept
- void **\_M\_profile\_resize** (std::size\_t \_\_old\_size)
- void **\_M\_swap** (\_Unordered\_profile &\_\_other) noexcept

#### Protected Attributes

- [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_info](#) \* **\_M\_hashfunc\_info**
- [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_info](#) \* **\_M\_size\_info**

#### 5.529.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class std::unordered\_multimap wrapper with performance instrumentation.

Definition at line 329 of file profile/unordered\_map.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

#### 5.530 std::\_\_profile::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inherits [unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc >, and [std::\\_\\_profile::\\_Unordered\\_profile](#)< \_Unordered<←  
Cont, \_Unique\_keys >.

## Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (const [unordered\\_multiset](#) &)=default
- **unordered\_multiset** (const [\\_Base](#) &\_\_x)
- **unordered\_multiset** ([unordered\\_multiset](#) &&)=default
- **unordered\_multiset** (const allocator\_type &\_\_a)
- **unordered\_multiset** (const [unordered\\_multiset](#) &\_\_umset, const allocator\_type &\_\_a)
- **unordered\_multiset** ([unordered\\_multiset](#) &&\_\_umset, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- `template<typename _InputIterator >`  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- `template<typename _InputIterator >`  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- void **clear** () noexcept
- `template<typename... _Args>`  
iterator **emplace** (\_Args &&... \_\_args)
- `template<typename... _Args>`  
iterator **emplace\_hint** (const\_iterator \_\_it, \_Args &&... \_\_args)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- iterator **insert** (value\_type &&\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, value\_type &&\_\_v)
- `template<typename _InputIter >`  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)



- `unordered_multiset` & `operator=` (const `unordered_multiset` &)=default
- `unordered_multiset` & `operator=` (`unordered_multiset` &&)=default
- `unordered_multiset` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- void `rehash` (`size_type` \_\_n)
- void `swap` (`unordered_multiset` &\_\_x) noexcept(noexcept(\_\_x.\_M\_base().swap(\_\_x)))

#### Protected Member Functions

- void `_M_profile_construct` () noexcept
- void `_M_profile_destruct` () noexcept
- void `_M_profile_resize` (std::size\_t \_\_old\_size)
- void `_M_swap` (\_Unordered\_profile &\_\_other) noexcept

#### Protected Attributes

- `__gnu_profile::__hashfunc_info` \* `_M_hashfunc_info`
- `__gnu_profile::__container_size_info` \* `_M_size_info`

#### 5.530.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc =
std::allocator<_Value>>
```

```
class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

`Unordered_multiset` wrapper with performance instrumentation.

Definition at line 307 of file `profile/unordered_set`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

#### 5.531 `std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >` Class Template Reference

Inherits `unordered_set< _Key, _Hash, _Pred, _Alloc >`, and `std::__profile::_Unordered_profile< _UnorderedCont, _Key, Unique_keys >`.

#### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const [unordered\\_set](#) &)=default
- **unordered\_set** (const [\\_Base](#) &\_\_x)
- **unordered\_set** ([unordered\\_set](#) &&)=default
- **unordered\_set** (const allocator\_type &\_\_a)
- **unordered\_set** (const [unordered\\_set](#) &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** ([unordered\\_set](#) &&\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- `template<typename _InputIterator >`  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- `template<typename _InputIterator >`  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- void **clear** () noexcept
- `template<typename... _Args >`  
[std::pair](#)< iterator, bool > **emplace** (\_Args &&... \_\_args)
- `template<typename... _Args >`  
iterator **emplace\_hint** (const\_iterator \_\_it, \_Args &&... \_\_args)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_obj)
- iterator **insert** (const\_iterator \_\_iter, value\_type &&\_\_v)
- `template<typename _InputIter >`  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- [unordered\\_set](#) & **operator=** (const [unordered\\_set](#) &)=default
- [unordered\\_set](#) & **operator=** ([unordered\\_set](#) &&)=default
- [unordered\\_set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_set](#) &\_\_x) noexcept(noexcept(\_\_x.\_M\_base().swap(\_\_x)))

## Protected Member Functions

- void **\_M\_profile\_construct** () noexcept
- void **\_M\_profile\_destruct** () noexcept
- void **\_M\_profile\_resize** (std::size\_t \_\_old\_size)
- void **\_M\_swap** (\_Unordered\_profile &\_\_other) noexcept

## Protected Attributes

- [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_info](#) \* [\\_M\\_hashfunc\\_info](#)
- [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_info](#) \* [\\_M\\_size\\_info](#)

## 5.531.1 Detailed Description

```
template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>>
class std::__profile::unordered_set<_Key, _Hash, _Pred, _Alloc>
```

Unordered\_set wrapper with performance instrumentation.

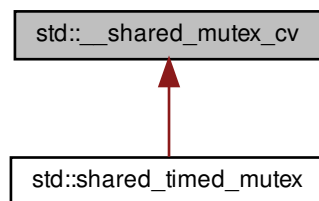
Definition at line 51 of file profile/unordered\_set.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 5.532 std::\_\_shared\_mutex\_cv Class Reference

Inheritance diagram for std::\_\_shared\_mutex\_cv:



## Public Member Functions

- [\\_\\_shared\\_mutex\\_cv](#) (const [\\_\\_shared\\_mutex\\_cv](#) &)=delete
- void [lock](#) ()
- void [lock\\_shared](#) ()
- [\\_\\_shared\\_mutex\\_cv](#) & [operator=](#) (const [\\_\\_shared\\_mutex\\_cv](#) &)=delete
- bool [try\\_lock](#) ()
- bool [try\\_lock\\_shared](#) ()
- void [unlock](#) ()
- void [unlock\\_shared](#) ()

## Friends

- class **shared\_timed\_mutex**

## 5.532.1 Detailed Description

A shared mutex type implemented using std::condition\_variable.

Definition at line 172 of file shared\_mutex.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 5.533 std::\_Base\_bitset&lt; \_Nw &gt; Struct Template Reference

## Public Types

- typedef unsigned long **\_WordT**

## Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
bool **\_M\_are\_all** () const noexcept
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- size\_t **\_M\_do\_count** () const noexcept
- size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept
- void **\_M\_do\_flip** () noexcept
- void **\_M\_do\_left\_shift** (size\_t \_\_shift) noexcept
- void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- void **\_M\_do\_reset** () noexcept
- void **\_M\_do\_right\_shift** (size\_t \_\_shift) noexcept
- void **\_M\_do\_set** () noexcept
- unsigned long long **\_M\_do\_to\_ullong** () const
- unsigned long **\_M\_do\_to\_ulong** () const
- void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- const \_WordT \* **\_M\_getdata** () const noexcept
- \_WordT & **\_M\_getword** (size\_t \_\_pos) noexcept
- constexpr \_WordT **\_M\_getword** (size\_t \_\_pos) const noexcept
- \_WordT & **\_M\_hiword** () noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- bool **\_M\_is\_any** () const noexcept
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) const noexcept

### Static Public Member Functions

- static constexpr `_WordT` **`_S_maskbit`** (`size_t __pos`) noexcept
- static constexpr `size_t` **`_S_whichbit`** (`size_t __pos`) noexcept
- static constexpr `size_t` **`_S_whichbyte`** (`size_t __pos`) noexcept
- static constexpr `size_t` **`_S_whichword`** (`size_t __pos`) noexcept

### Public Attributes

- `_WordT` **`_M_w`** [`_Nw`]

#### 5.533.1 Detailed Description

```
template<size_t _Nw>
struct std::_Base_bitset< _Nw >
```

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.

See documentation for `bitset`.

Definition at line 75 of file `bitset`.

#### 5.533.2 Member Data Documentation

##### 5.533.2.1 `_M_w`

```
template<size_t _Nw>
_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]
```

0 is the least significant word.

Definition at line 80 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

#### 5.534 `std::_Base_bitset< 0 >` Struct Template Reference

### Public Types

- typedef unsigned long **`_WordT`**

## Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long) noexcept
- template<size\_t \_Nb>  
bool **\_M\_are\_all** () const noexcept
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)< 0 > &) noexcept
- size\_t **\_M\_do\_count** () const noexcept
- size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept
- void **\_M\_do\_flip** () noexcept
- void **\_M\_do\_left\_shift** (size\_t) noexcept
- void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< 0 > &) noexcept
- void **\_M\_do\_reset** () noexcept
- void **\_M\_do\_right\_shift** (size\_t) noexcept
- void **\_M\_do\_set** () noexcept
- unsigned long long **\_M\_do\_to\_ullong** () const noexcept
- unsigned long **\_M\_do\_to\_ulong** () const noexcept
- void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< 0 > &) noexcept
- \_WordT & **\_M\_getword** (size\_t) noexcept
- constexpr \_WordT **\_M\_getword** (size\_t) const noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- bool **\_M\_is\_any** () const noexcept
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< 0 > &) const noexcept

## Static Public Member Functions

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept

## 5.534.1 Detailed Description

```
template<>
struct std::_Base_bitset< 0 >
```

Base class, specialization for no storage (zero-length bitset).

See documentation for bitset.

Definition at line 523 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.535 std::\_Base\_bitset< 1 > Struct Template Reference

### Public Types

- typedef unsigned long **\_WordT**

### Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
bool **\_M\_are\_all** () const noexcept
- void **\_M\_do\_and** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) noexcept
- size\_t **\_M\_do\_count** () const noexcept
- size\_t **\_M\_do\_find\_first** (size\_t \_\_not\_found) const noexcept
- size\_t **\_M\_do\_find\_next** (size\_t \_\_prev, size\_t \_\_not\_found) const noexcept
- void **\_M\_do\_flip** () noexcept
- void **\_M\_do\_left\_shift** (size\_t \_\_shift) noexcept
- void **\_M\_do\_or** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) noexcept
- void **\_M\_do\_reset** () noexcept
- void **\_M\_do\_right\_shift** (size\_t \_\_shift) noexcept
- void **\_M\_do\_set** () noexcept
- unsigned long long **\_M\_do\_to\_ullong** () const noexcept
- unsigned long **\_M\_do\_to\_ulong** () const noexcept
- void **\_M\_do\_xor** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) noexcept
- const \_WordT \* **\_M\_getdata** () const noexcept
- \_WordT & **\_M\_getword** (size\_t) noexcept
- constexpr \_WordT **\_M\_getword** (size\_t) const noexcept
- \_WordT & **\_M\_hiword** () noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- bool **\_M\_is\_any** () const noexcept
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) const noexcept

### Static Public Member Functions

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept

### Public Attributes

- \_WordT **\_M\_w**

## 5.535.1 Detailed Description

```
template<>
struct std::_Base_bitset< 1 >
```

Base class, specialization for a single word.

See documentation for `bitset`.

Definition at line 376 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.536 `std::_Bind<_Signature>` Struct Template Reference

## 5.536.1 Detailed Description

```
template<typename _Signature>
struct std::_Bind<_Signature>
```

Type of the function object returned from `bind()`.

Definition at line 383 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.537 `std::_Bind_result<_Result, _Signature>` Struct Template Reference

## 5.537.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::_Bind_result<_Result, _Signature>
```

Type of the function object returned from `bind<R>()`.

Definition at line 531 of file `functional`.

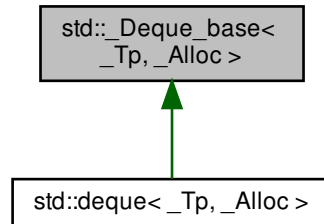
The documentation for this struct was generated from the following file:

- [functional](#)



### 5.538 `std::_Deque_base<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::_Deque_base<_Tp, _Alloc>`:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Deque_iterator<_Tp, const _Tp &, _Ptr_const>` **const\_iterator**
- typedef `_Deque_iterator<_Tp, _Tp &, _Ptr>` **iterator**
- typedef `_Alloc_traits::size_type` **size\_type**

#### Public Member Functions

- `_Deque_base` (`size_t __num_elements`)
- `_Deque_base` (`const allocator_type &__a, size_t __num_elements`)
- `_Deque_base` (`const allocator_type &__a`)
- `_Deque_base` (`_Deque_base &&__x, false_type`)
- `_Deque_base` (`_Deque_base &&__x, true_type`)
- `_Deque_base` (`_Deque_base &&__x`)
- `_Deque_base` (`_Deque_base &&__x, const allocator_type &__a, size_type __n`)
- `allocator_type` **get\_allocator** () const noexcept

#### Protected Types

- enum { `_S_initial_map_size` }
- typedef `_gnu_cxx::__alloc_traits<_Tp_alloc_type>` **\_Alloc\_traits**
- typedef `_gnu_cxx::__alloc_traits<_Map_alloc_type>` **\_Map\_alloc\_traits**
- typedef `_Alloc_traits::template rebind<_Ptr>::other` **\_Map\_alloc\_type**
- typedef `iterator::_Map_pointer` **\_Map\_pointer**
- typedef `_Alloc_traits::pointer` **\_Ptr**
- typedef `_Alloc_traits::const_pointer` **\_Ptr\_const**
- typedef `_gnu_cxx::__alloc_traits<_Alloc>::template rebind<_Tp>::other` **\_Tp\_alloc\_type**

## Protected Member Functions

- `_Map_pointer _M_allocate_map (size_t __n)`
- `_Ptr _M_allocate_node ()`
- `void _M_create_nodes (_Map_pointer __nstart, _Map_pointer __nfinish)`
- `void _M_deallocate_map (_Map_pointer __p, size_t __n) noexcept`
- `void _M_deallocate_node (_Ptr __p) noexcept`
- `void _M_destroy_nodes (_Map_pointer __nstart, _Map_pointer __nfinish) noexcept`
- `_Map_alloc_type _M_get_map_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `void _M_initialize_map (size_t)`

## Protected Attributes

- `_Deque_impl _M_impl`

## 5.538.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::Deque_base< _Tp, _Alloc >
```

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual `Tp` element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 461 of file `stl_deque.h`.

## 5.538.2 Member Function Documentation

5.538.2.1 `_M_initialize_map()`

```
template<typename _Tp , typename _Alloc >
void std::Deque_base< _Tp, _Alloc >::_M_initialize_map (
    size_t __num_elements ) [protected]
```

Layout storage.

## Parameters

<code>__num_elements</code>	The count of <code>T</code> 's for which to allocate space at first.
-----------------------------	--

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 683 of file `stl_deque.h`.

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

**5.539 std::\_Deque\_iterator< \_Tp, \_Ref, \_Ptr > Struct Template Reference****Public Types**

- typedef \_\_ptr\_to< \_Tp > **\_Elt\_pointer**
- typedef \_\_ptr\_to< \_Elt\_pointer > **\_Map\_pointer**
- typedef [\\_Deque\\_iterator](#) **\_Self**
- typedef \_\_iter< const \_Tp > **const\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef \_\_iter< \_Tp > **iterator**
- typedef [std::random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Ptr **pointer**
- typedef \_Ref **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

**Public Member Functions**

- **\_Deque\_iterator** (\_Elt\_pointer \_\_x, \_Map\_pointer \_\_y) noexcept
- **\_Deque\_iterator** (const [iterator](#) &\_\_x) noexcept
- [iterator](#) **\_M\_const\_cast** () const noexcept
- void **\_M\_set\_node** (\_Map\_pointer \_\_new\_node) noexcept
- reference **operator\*** () const noexcept
- **\_Self** **operator+** (difference\_type \_\_n) const noexcept
- **\_Self** & **operator++** () noexcept
- **\_Self** **operator++** (int) noexcept
- **\_Self** & **operator+=** (difference\_type \_\_n) noexcept
- **\_Self** **operator-** (difference\_type \_\_n) const noexcept
- **\_Self** & **operator--** () noexcept
- **\_Self** **operator--** (int) noexcept
- **\_Self** & **operator-=** (difference\_type \_\_n) noexcept
- pointer **operator->** () const noexcept
- reference **operator[]** (difference\_type \_\_n) const noexcept

**Static Public Member Functions**

- static size\_t **\_S\_buffer\_size** () noexcept

## Public Attributes

- `_Elt_pointer _M_cur`
- `_Elt_pointer _M_first`
- `_Elt_pointer _M_last`
- `_Map_pointer _M_node`

### 5.539.1 Detailed Description

```
template<typename _Tp, typename _Ref, typename _Ptr>
struct std::_Deque_iterator< _Tp, _Ref, _Ptr >
```

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 109 of file `stl_deque.h`.

### 5.539.2 Member Function Documentation

#### 5.539.2.1 `_M_set_node()`

```
template<typename _Tp, typename _Ref, typename _Ptr>
void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (
    _Map_pointer __new_node ) [inline], [noexcept]
```

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

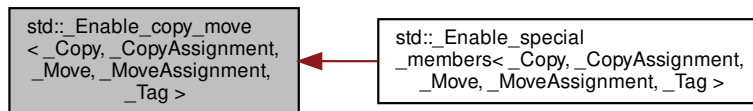
Definition at line 255 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl\\_deque.h](#)

### 5.540 `std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >` Struct Template Reference

Inheritance diagram for `std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`:



#### 5.540.1 Detailed Description

```
template<bool _Copy, bool _CopyAssignment, bool _Move, bool _MoveAssignment, typename _Tag = void>
struct std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >
```

A mixin helper to conditionally enable or disable the copy/move special members.

See also

[`\_Enable\_special\_members`](#)

Definition at line 84 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

### 5.541 `std::_Enable_default_constructor< _Switch, _Tag >` Struct Template Reference

#### Public Member Functions

- `constexpr` [`\_Enable\_default\_constructor`](#) ([`\_Enable\_default\_constructor`](#) const &) noexcept=default
- `constexpr` [`\_Enable\_default\_constructor`](#) ([`\_Enable\_default\_constructor`](#) &&) noexcept=default
- `constexpr` [`\_Enable\_default\_constructor`](#) ([`\_Enable\_default\_constructor\_tag`](#))
- [`\_Enable\_default\_constructor`](#) & `operator=` ([`\_Enable\_default\_constructor`](#) const &) noexcept=default
- [`\_Enable\_default\_constructor`](#) & `operator=` ([`\_Enable\_default\_constructor`](#) &&) noexcept=default

## 5.541.1 Detailed Description

```
template<bool _Switch, typename _Tag = void>
struct std::_Enable_default_constructor<_Switch,_Tag>
```

A mixin helper to conditionally enable or disable the default constructor.

See also

`_Enable_special_members`

Definition at line 50 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

5.542 `std::_Enable_destructor<_Switch,_Tag>` Struct Template Reference

## 5.542.1 Detailed Description

```
template<bool _Switch, typename _Tag = void>
struct std::_Enable_destructor<_Switch,_Tag>
```

A mixin helper to conditionally enable or disable the default destructor.

See also

`_Enable_special_members`

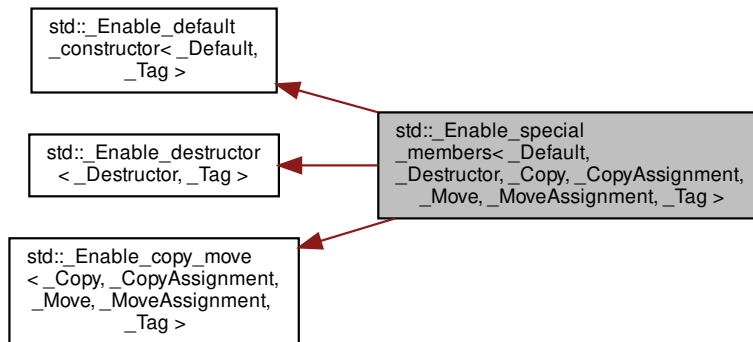
Definition at line 74 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

### 5.543 `std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >` Struct Template Reference

Inheritance diagram for `std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`:



#### 5.543.1 Detailed Description

```

template<bool _Default, bool _Destructor, bool _Copy, bool _CopyAssignment, bool _Move, bool _MoveAssignment, typename _Tag = void>
struct std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >

```

A mixin helper to conditionally enable or disable the special members.

The `_Tag` type parameter is to make mixin bases unique and thus avoid ambiguities.

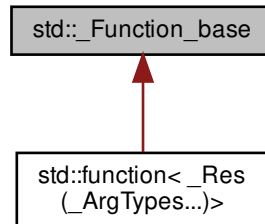
Definition at line 97 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

## 5.544 std::\_Function\_base Class Reference

Inheritance diagram for std::\_Function\_base:



### Public Types

- `typedef bool(* _Manager_type) (_Any_data &, const _Any_data &, _Manager_operation)`

### Public Member Functions

- `bool _M_empty () const`

### Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

### Static Public Attributes

- `static const std::size_t _M_max_align`
- `static const std::size_t _M_max_size`

#### 5.544.1 Detailed Description

Base class of all polymorphic function object wrappers.

Definition at line 131 of file `std_function.h`.

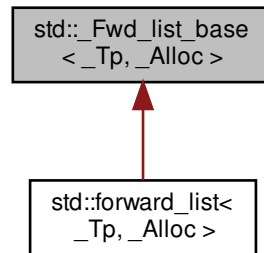
The documentation for this class was generated from the following file:

- [std\\_function.h](#)



## 5.545 std::\_Fwd\_list\_base<\_Tp, \_Alloc> Struct Template Reference

Inheritance diagram for std::\_Fwd\_list\_base<\_Tp, \_Alloc>:



### Public Types

- typedef `_Fwd_list_node<_Tp>` `_Node`
- typedef `_Fwd_list_const_iterator<_Tp>` `const_iterator`
- typedef `_Fwd_list_iterator<_Tp>` `iterator`

### Public Member Functions

- `_Fwd_list_base` (`_Node_alloc_type` &&\_\_a)
- `_Fwd_list_base` (`_Fwd_list_base` &&\_\_lst, `_Node_alloc_type` &&\_\_a, `std::true_type`)
- `_Fwd_list_base` (`_Fwd_list_base` &&\_\_lst, `_Node_alloc_type` &&\_\_a)
- `_Fwd_list_base` (`_Fwd_list_base` &&)=default
- `_Node_alloc_type` & `_M_get_Node_allocator` () noexcept
- `const _Node_alloc_type` & `_M_get_Node_allocator` () const noexcept

### Protected Types

- typedef `__gnu_cxx::__alloc_traits<_Node_alloc_type>` `_Node_alloc_traits`
- typedef `__alloc_rebind<_Alloc, _Fwd_list_node<_Tp>>` `_Node_alloc_type`

### Protected Member Functions

- template<typename... \_Args>  
`_Node` \* `_M_create_node` (`_Args` &&... \_\_args)
- `_Fwd_list_node_base` \* `_M_erase_after` (`_Fwd_list_node_base` \*\_\_pos)
- `_Fwd_list_node_base` \* `_M_erase_after` (`_Fwd_list_node_base` \*\_\_pos, `_Fwd_list_node_base` \*\_\_last)
- `_Node` \* `_M_get_node` ()
- template<typename... \_Args>  
`_Fwd_list_node_base` \* `_M_insert_after` (`const_iterator` \_\_pos, `_Args` &&... \_\_args)
- void `_M_put_node` (`_Node` \*\_\_p)

## Protected Attributes

- `_Fwd_list_impl _M_impl`

## 5.545.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::_Fwd_list_base< _Tp, _Alloc >
```

Base class for forward\_list.

Definition at line 289 of file forward\_list.h.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.546 std::\_Fwd\_list\_const\_iterator&lt; \_Tp &gt; Struct Template Reference

## Public Types

- typedef const [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_const\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **iterator**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

## Public Member Functions

- **\_Fwd\_list\_const\_iterator** (const [\\_Fwd\\_list\\_node\\_base](#) \* \_\_n) noexcept
- **\_Fwd\_list\_const\_iterator** (const [iterator](#) & \_\_iter) noexcept
- **\_Self \_M\_next** () const noexcept
- bool **operator!=** (const [\\_Self](#) & \_\_x) const noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept
- bool **operator==** (const [\\_Self](#) & \_\_x) const noexcept

## Public Attributes

- const [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_node**

#### 5.546.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_const_iterator< _Tp >
```

A forward\_list::const\_iterator.

All the functions are op overloads.

Definition at line 202 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

#### 5.547 std::\_Fwd\_list\_iterator< \_Tp > Struct Template Reference

##### Public Types

- typedef [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

##### Public Member Functions

- [\\_Fwd\\_list\\_iterator](#) ([\\_Fwd\\_list\\_node\\_base](#) \*\_\_n) noexcept
- [\\_Self \\_M\\_next](#) () const noexcept
- bool **operator!=** (const [\\_Self](#) &\_\_x) const noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept
- bool **operator==** (const [\\_Self](#) &\_\_x) const noexcept

##### Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_node**

## 5.547.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_iterator< _Tp >
```

A forward\_list::iterator.

All the functions are op overloads.

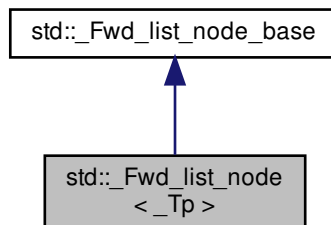
Definition at line 135 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.548 std::\_Fwd\_list\_node&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



## Public Member Functions

- void **\_M\_reverse\_after** () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_transfer\_after** ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end) noexcept
- [\\_Tp](#) \* **\_M\_valptr** () noexcept
- const [\\_Tp](#) \* **\_M\_valptr** () const noexcept

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_next**
- [\\_\\_gnu\\_cxx::\\_\\_aligned\\_buffer< \\_Tp >](#) **\_M\_storage**

### 5.548.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_node< _Tp >
```

A helper node class for `forward_list`. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

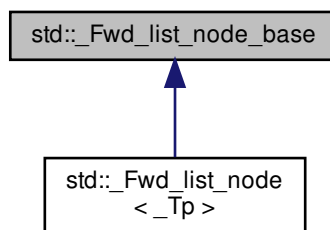
Definition at line 113 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 5.549 std::\_Fwd\_list\_node\_base Struct Reference

Inheritance diagram for `std::_Fwd_list_node_base`:



#### Public Member Functions

- `_Fwd_list_node_base` (`_Fwd_list_node_base` &&\_\_x) noexcept
- `_Fwd_list_node_base` (const `_Fwd_list_node_base` &)=delete
- `void _M_reverse_after` () noexcept
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base` \*\_\_begin, `_Fwd_list_node_base` \*\_\_end) noexcept
- `_Fwd_list_node_base` & `operator=` (const `_Fwd_list_node_base` &)=delete
- `_Fwd_list_node_base` & `operator=` (`_Fwd_list_node_base` &&\_\_x) noexcept

#### Public Attributes

- `_Fwd_list_node_base` \* `_M_next`

### 5.549.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

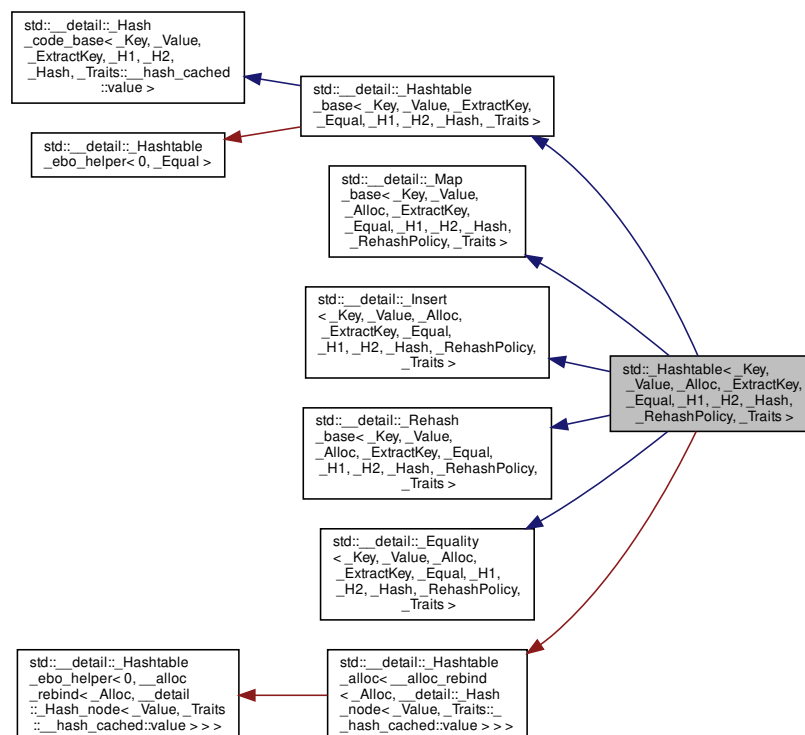
Definition at line 54 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 5.550 `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Class Template Reference

Inheritance diagram for `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- using **const\_iterator** = typename `__hashtable_base::const_iterator`
- using **const\_local\_iterator** = typename `__hashtable_base::const_local_iterator`
- typedef `__value_alloc_traits::const_pointer` **const\_pointer**
- typedef const value\_type & **const\_reference**
- using **difference\_type** = typename `__hashtable_base::difference_type`
- using **iterator** = typename `__hashtable_base::iterator`
- typedef `_Equal` **key\_equal**
- typedef `_Key` **key\_type**
- using **local\_iterator** = typename `__hashtable_base::local_iterator`
- typedef `__value_alloc_traits::pointer` **pointer**
- typedef value\_type & **reference**
- using **size\_type** = typename `__hashtable_base::size_type`
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Hashtable** (size\_type \_\_bucket\_hint, const \_H1 &, const \_H2 &, const \_Hash &, const \_Equal &, const \_ExtractKey &, const allocator\_type &)
- template<typename \_InputIterator >  
**\_Hashtable** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_bucket\_hint, const \_H1 &, const \_H2 &, const \_Hash &, const \_Equal &, const \_ExtractKey &, const allocator\_type &)
- **\_Hashtable** (const [\\_Hashtable](#) &)
- **\_Hashtable** ([\\_Hashtable](#) &&) noexcept
- **\_Hashtable** (const [\\_Hashtable](#) &, const allocator\_type &)
- **\_Hashtable** ([\\_Hashtable](#) &&, const allocator\_type &)
- **\_Hashtable** (const allocator\_type & \_\_a)
- **\_Hashtable** (size\_type \_\_n, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- template<typename \_InputIterator >  
**\_Hashtable** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- **\_Hashtable** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- const \_RehashPolicy & **\_\_rehash\_policy** () const
- void **\_\_rehash\_policy** (const \_RehashPolicy & \_\_pol)
- template<typename... \_Args>  
auto **\_M\_emplace** (std::true\_type, \_Args &&... \_\_args) -> [pair](#)< iterator, bool >
- template<typename... \_Args>  
auto **\_M\_emplace** (const\_iterator \_\_hint, std::false\_type, \_Args &&... \_\_args) -> iterator
- template<typename \_Arg, typename \_NodeGenerator >  
auto **\_M\_insert** (\_Arg && \_\_v, const \_NodeGenerator & \_\_node\_gen, true\_type, size\_type \_\_n\_elt) -> [pair](#)< iterator, bool >
- template<typename \_Arg, typename \_NodeGenerator >  
auto **\_M\_insert** (const\_iterator \_\_hint, \_Arg && \_\_v, const \_NodeGenerator & \_\_node\_gen, false\_type) -> iterator
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- local\_iterator **begin** (size\_type \_\_n)
- const\_local\_iterator **begin** (size\_type \_\_n) const

- `size_type bucket` (const key\_type &\_\_k) const
- `size_type bucket_count` () const noexcept
- `size_type bucket_size` (size\_type \_\_n) const
- `const_iterator cbegin` () const noexcept
- `const_local_iterator cbegin` (size\_type \_\_n) const
- `const_iterator cend` () const noexcept
- `const_local_iterator cend` (size\_type \_\_n) const
- `void clear` () noexcept
- `size_type count` (const key\_type &\_\_k) const
- `template<typename... _Args>`  
`__ireturn_type emplace` (\_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator emplace_hint` (const\_iterator \_\_hint, \_Args &&... \_\_args)
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `local_iterator end` (size\_type \_\_n)
- `const_local_iterator end` (size\_type \_\_n) const
- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_k)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_k) const
- `iterator erase` (const\_iterator)
- `iterator erase` (iterator \_\_it)
- `size_type erase` (const key\_type &\_\_k)
- `iterator erase` (const\_iterator, const\_iterator)
- `iterator find` (const key\_type &\_\_k)
- `const_iterator find` (const key\_type &\_\_k) const
- `allocator_type get_allocator` () const noexcept
- `key_equal key_eq` () const
- `float load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- `size_type max_size` () const noexcept
- `_Hashtable & operator=` (const \_Hashtable &\_\_ht)
- `_Hashtable & operator=` (\_Hashtable &&\_\_ht) noexcept(\_\_node\_alloc\_traits::\_S\_nothrow\_move() &&is\_nothrow\_move\_assignable<\_H1 >::value &&is\_nothrow\_move\_assignable<\_Equal >::value)
- `_Hashtable & operator=` (initializer\_list< value\_type > \_\_l)
- `void rehash` (size\_type \_\_n)
- `size_type size` () const noexcept
- `void swap` (\_Hashtable &) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_H1 >, \_\_is\_nothrow\_swappable<\_Equal >>::value)

#### Protected Member Functions

- `size_type _M_bucket_index` (\_\_node\_type \*\_\_n) const noexcept
- `size_type _M_bucket_index` (const key\_type &\_\_k, \_\_hash\_code \_\_c) const
- `template<typename... _Args>`  
`std::pair< iterator, bool > _M_emplace` (std::true\_type, \_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator _M_emplace` (std::false\_type \_\_uk, \_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator _M_emplace` (const\_iterator, std::true\_type \_\_uk, \_Args &&... \_\_args)



- `template<typename... _Args>`  
`iterator _M_emplace (const_iterator, std::false\_type, _Args &&... __args)`
- `const _Equal & _M_eq () const`
- `_Equal & _M_eq ()`
- `bool _M_equals (const _Key &__k, __hash_code __c, \_\_node\_type *__n) const`
- `size_type _M_erase (std::true\_type, const key_type &)`
- `size_type _M_erase (std::false\_type, const key_type &)`
- `iterator _M_erase (size_type __bkt, __node_base *__prev_n, \_\_node\_type *__n)`
- `__node_base * _M_find_before_node (size_type, const key_type &, __hash_code) const`
- `\_\_node\_type * _M_find_node (size_type __bkt, const key_type &__key, __hash_code __c) const`
- `__node_base * _M_get_previous_node (size_type __bkt, __node_base *__n)`
- `template<typename _Arg, typename _NodeGenerator >`  
`std::pair< iterator, bool > _M_insert (_Arg &&, const _NodeGenerator &, true\_type, size_type=1)`
- `template<typename _Arg, typename _NodeGenerator >`  
`iterator _M_insert (_Arg &&__arg, const _NodeGenerator &__node_gen, false\_type __uk)`
- `template<typename _Arg, typename _NodeGenerator >`  
`iterator _M_insert (const_iterator, _Arg &&__arg, const _NodeGenerator &__node_gen, true\_type __uk)`
- `template<typename _Arg, typename _NodeGenerator >`  
`iterator _M_insert (const_iterator, _Arg &&, const _NodeGenerator &, false\_type)`
- `void _M_insert_bucket_begin (size_type, \_\_node\_type *)`
- `iterator _M_insert_multi_node (\_\_node\_type *__hint, __hash_code __code, \_\_node\_type *__n)`
- `iterator _M_insert_unique_node (size_type __bkt, __hash_code __code, \_\_node\_type *__n, size_type __n←  
__elt=1)`
- `void _M_remove_bucket_begin (size_type __bkt, \_\_node\_type *__next_n, size_type __next_bkt)`
- `void _M_swap (_Hashtable_base &__x)`

### Private Types

- using `__bucket_alloc_traits` = `std::allocator\_traits< __bucket_alloc_type >`
- using `__bucket_alloc_type` = `__alloc_rebind< __node_alloc_type, __bucket_type >`

### Private Member Functions

- `__bucket_type * _M_allocate_buckets (std::size_t __n)`
- `\_\_node\_type * _M_allocate_node (_Args &&... __args)`
- `void _M_deallocate_buckets (__bucket_type *, std::size_t __n)`
- `void _M_deallocate_node (\_\_node\_type *__n)`
- `void _M_deallocate_nodes (\_\_node\_type *__n)`
- `__node_alloc_type & _M_node_allocator ()`
- `const __node_alloc_type & _M_node_allocator () const`

### Friends

- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Constant_iteratorsa>`  
`struct \_\_detail::Insert`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa >`  
`struct \_\_detail::Insert\_base`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Unique_keysa>`  
`struct \_\_detail::Map\_base`

### 5.550.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
class std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template \_Hashtable.

#### Template Parameters

<code>_Value</code>	CopyConstructible type.
<code>_Key</code>	CopyConstructible type.
<code>_Alloc</code>	An allocator type ([lib.allocator.requirements]) whose <code>_Alloc::value_type</code> is <code>_Value</code> . As a conforming extension, we allow for <code>_Alloc::value_type != _Value</code> .
<code>_ExtractKey</code>	Function object that takes an object of type <code>_Value</code> and returns a value of type <code>_Key</code> .
<code>_Equal</code>	Function object that takes two objects of type <code>k</code> and returns a bool-like value that is true if the two objects are considered equal.
<code>_H1</code>	The hash function. A unary function object with argument type <code>_Key</code> and result type <code>size_t</code> . Return values should be distributed over the entire range <code>[0, numeric_limits&lt;size_t&gt;::max())</code> .
<code>_H2</code>	The range-hashing function (in the terminology of Tavori and Dreizin). A binary function object whose argument types and result type are all <code>size_t</code> . Given arguments <code>r</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> .
<code>_Hash</code>	The ranged hash function (Tavori and Dreizin). A binary function whose argument types are <code>_Key</code> and <code>size_t</code> and whose result type is <code>size_t</code> . Given arguments <code>k</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> . Default: <code>hash(k, N) = h2(h1(k), N)</code> . If <code>_Hash</code> is anything other than the default, <code>_H1</code> and <code>_H2</code> are ignored.
<code>_RehashPolicy</code>	Policy class with three members, all of which govern the bucket count. <code>_M_next_bkt(n)</code> returns a bucket count no smaller than <code>n</code> . <code>_M_bkt_for_elements(n)</code> returns a bucket count appropriate for an element count of <code>n</code> . <code>_M_need_rehash(n_bkt, n_elt, n_ins)</code> determines whether, if the current bucket count is <code>n_bkt</code> and the current element count is <code>n_elt</code> , we need to increase the bucket count. If so, returns <code>make_pair(true, n)</code> , where <code>n</code> is the new bucket count. If not, returns <code>make_pair(false, &lt;anything&gt;)</code>
<code>_Traits</code>	Compile-time class with three boolean <code>std::integral_constant</code> members: <code>__cache_hash_code</code> , <code>__constant_iterators</code> , <code>__unique_keys</code> .

Each \_Hashtable data structure has:

- `_Bucket[] _M_buckets`
- `_Hash_node_base _M_before_begin`
- `size_type _M_bucket_count`
- `size_type _M_element_count`

with `_Bucket` being `_Hash_node*` and `_Hash_node` containing:

- `_Hash_node* _M_next`

- `Tp _M_value`
- `size_t _M_hash_code` if `cache_hash_code` is true

In terms of Standard containers the hashtable is like the aggregation of:

- `std::forward_list<_Node>` containing the elements
- `std::vector<std::forward_list<_Node>::iterator>` representing the buckets

The non-empty buckets contain the node before the first node in the bucket. This design makes it possible to implement something like `std::forward_list::insert_after` on container insertion and `std::forward_list::erase_after` on container erase calls. `_M_before_begin` is equivalent to `std::forward_list::before_begin`. Empty buckets contain `nullptr`. Note that one of the non-empty buckets contains `&_M_before_begin` which is not a dereferenceable node so the node pointer in a bucket shall never be dereferenced, only its next node can be.

Walking through a bucket's nodes requires a check on the hash code to see if each node is still in the bucket. Such a design assumes a quite efficient hash functor and is one of the reasons it is highly advisable to set `__cache_hash_code` to true.

The container iterators are simply built from nodes. This way incrementing the iterator is perfectly efficient independent of how many empty buckets there are in the container.

On insert we compute the element's hash code and use it to find the bucket index. If the element must be inserted in an empty bucket we add it at the beginning of the singly linked list and make the bucket point to `_M_before_begin`. The bucket that used to point to `_M_before_begin`, if any, is updated to point to its new before begin node.

On erase, the simple iterator design requires using the hash functor to get the index of the bucket to update. For this reason, when `__cache_hash_code` is set to false the hash functor must not throw and this is enforced by a static assertion.

Functionality is implemented by decomposition into base classes, where the derived `_Hashtable` class is used in `_Map_base`, `_Insert`, `_Rehash_base`, and `_Equality` base classes to access the "this" pointer. `_Hashtable_base` is used in the base classes as a non-recursive, fully-completed-type so that detailed nested type information, such as iterator type and node type, can be used. This is similar to the "Curiously Recurring Template Pattern" (CRTP) technique, but uses a reconstructed, not explicitly passed, template pattern.

Base class templates are:

- `__detail::_Hashtable_base`
- `__detail::_Map_base`
- `__detail::_Insert`
- `__detail::_Rehash_base`
- `__detail::_Equality`

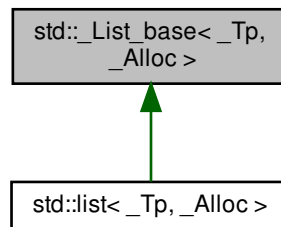
Definition at line 173 of file `bits/hashtable.h`.

The documentation for this class was generated from the following file:

- [bits/hashtable.h](#)

## 5.551 std::\_List\_base&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_List\_base< \_Tp, \_Alloc >:



## Public Types

- typedef \_Alloc **allocator\_type**

## Public Member Functions

- **\_List\_base** (const \_Node\_alloc\_type &\_\_a) noexcept
- **\_List\_base** (\_List\_base &&)=default
- **\_List\_base** (\_List\_base &&\_\_x, \_Node\_alloc\_type &&\_\_a)
- **\_List\_base** (\_Node\_alloc\_type &&\_\_a, \_List\_base &&\_\_x)
- **\_List\_base** (\_Node\_alloc\_type &&\_\_a)
- void **\_M\_clear** () noexcept
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept
- void **\_M\_init** () noexcept
- void **\_M\_move\_nodes** (\_List\_base &&\_\_x)

## Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Node\_alloc\_type > **\_Node\_alloc\_traits**
- typedef [\\_Tp\\_alloc\\_traits](#)::template rebind< [\\_List\\_node](#)< \_Tp > >::other **\_Node\_alloc\_type**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Tp\_alloc\_type > **\_Tp\_alloc\_traits**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc >::template rebind< \_Tp >::other **\_Tp\_alloc\_type**

### Protected Member Functions

- void **\_M\_dec\_size** (size\_t)
- size\_t **\_M\_distance** (const void \*, const void \*) const
- \_Node\_alloc\_traits::pointer **\_M\_get\_node** ()
- size\_t **\_M\_get\_size** () const
- void **\_M\_inc\_size** (size\_t)
- size\_t **\_M\_node\_count** () const
- void **\_M\_put\_node** (typename \_Node\_alloc\_traits::pointer \_\_p) noexcept
- void **\_M\_set\_size** (size\_t)

### Static Protected Member Functions

- static size\_t **\_S\_distance** (const [\\_\\_detail::\\_List\\_node\\_base](#) \* \_\_first, const [\\_\\_detail::\\_List\\_node\\_base](#) \* \_\_last)

### Protected Attributes

- [\\_List\\_impl](#) **\_M\_impl**

#### 5.551.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_List_base< _Tp, _Alloc >
```

See bits/stl\_deque.h's `_Deque_base` for an explanation.

Definition at line 357 of file `stl_list.h`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

#### 5.552 `std::_List_const_iterator< _Tp >` Struct Template Reference

##### Public Types

- typedef const [\\_List\\_node](#)< \_Tp > **\_Node**
- typedef [\\_List\\_const\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [\\_List\\_iterator](#)< \_Tp > **iterator**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

## Public Member Functions

- `_List_const_iterator` (const `__detail::_List_node_base * __x`) noexcept
- `_List_const_iterator` (const `iterator & __x`) noexcept
- `iterator _M_const_cast` () const noexcept
- `bool operator!=` (const `_Self & __x`) const noexcept
- reference `operator*` () const noexcept
- `_Self & operator++` () noexcept
- `_Self operator++` (int) noexcept
- `_Self & operator--` () noexcept
- `_Self operator--` (int) noexcept
- pointer `operator->` () const noexcept
- `bool operator==` (const `_Self & __x`) const noexcept

## Public Attributes

- const `__detail::_List_node_base * _M_node`

## 5.552.1 Detailed Description

```
template<typename _Tp>
struct std::_List_const_iterator<_Tp>
```

A `list::const_iterator`.

All the functions are op overloads.

Definition at line 74 of file `stl_iterator_base_funcs.h`.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

5.553 `std::_List_iterator<_Tp>` Struct Template Reference

## Public Types

- `typedef _List_node<_Tp> _Node`
- `typedef _List_iterator<_Tp> _Self`
- `typedef ptrdiff_t difference_type`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef _Tp value_type`

## Public Member Functions

- [\\_List\\_iterator](#) ([\\_\\_detail::\\_List\\_node\\_base](#) \* \_\_x) noexcept
- [\\_Self \\_M\\_const\\_cast](#) () const noexcept
- bool **operator!=** (const [\\_Self](#) & \_\_x) const noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- [\\_Self](#) & **operator--** () noexcept
- [\\_Self](#) **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- bool **operator==** (const [\\_Self](#) & \_\_x) const noexcept

## Public Attributes

- [\\_\\_detail::\\_List\\_node\\_base](#) \* **\_M\_node**

## 5.553.1 Detailed Description

```
template<typename _Tp>
struct std::_List_iterator< _Tp >
```

A list::iterator.

All the functions are op overloads.

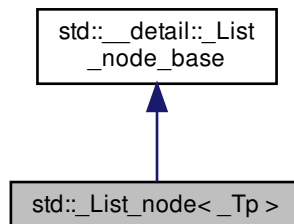
Definition at line 73 of file `stl_iterator_base_funcs.h`.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

5.554 `std::_List_node< _Tp >` Struct Template Reference

Inheritance diagram for `std::_List_node< _Tp >`:



## Public Member Functions

- void **\_M\_hook** (\_List\_node\_base \*const \_\_position) noexcept
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** (\_List\_node\_base \*const \_\_first, \_List\_node\_base \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept
- \_Tp \* **\_M\_valptr** ()
- \_Tp const \* **\_M\_valptr** () const

## Static Public Member Functions

- static void **swap** (\_List\_node\_base &\_\_x, \_List\_node\_base &\_\_y) noexcept

## Public Attributes

- \_List\_node\_base \* **\_M\_next**
- \_List\_node\_base \* **\_M\_prev**
- \_\_gnu\_cxx::\_\_aligned\_membuf< \_Tp > **\_M\_storage**

## 5.554.1 Detailed Description

```
template<typename _Tp>
struct std::_List_node< _Tp >
```

An actual node in the list.

Definition at line 166 of file stl\_list.h.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.555 std::\_Maybe\_get\_result\_type&lt; \_Functor, typename &gt; Struct Template Reference

## 5.555.1 Detailed Description

```
template<typename _Functor, typename = __void_t<>>
struct std::_Maybe_get_result_type< _Functor, typename >
```

If we have found a result\_type, extract it.

Definition at line 112 of file refwrap.h.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)



## 5.556 `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes>` Struct Template Reference

### 5.556.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
struct std::_Maybe_unary_or_binary_function<_Res, _ArgTypes>
```

Derives from `unary_function` or `binary_function`, or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

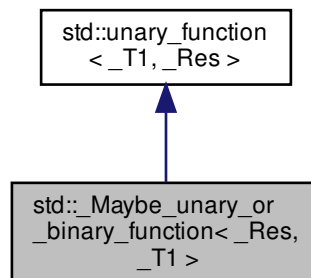
Definition at line 53 of file `refwrap.h`.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

## 5.557 `std::_Maybe_unary_or_binary_function<_Res, _T1>` Struct Template Reference

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _T1>`:



### Public Types

- typedef `_T1` [argument\\_type](#)
- typedef `_Res` [result\\_type](#)

### 5.557.1 Detailed Description

```
template<typename _Res, typename _T1>
struct std::_Maybe_unary_or_binary_function<_Res, _T1>
```

Derives from `unary_function`, as appropriate.

Definition at line 57 of file `refwrap.h`.

## 5.557.2 Member Typedef Documentation

## 5.557.2.1 argument\_type

```
typedef _T1 std::unary_function< _T1 , _Res >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 5.557.2.2 result\_type

```
typedef _Res std::unary_function< _T1 , _Res >::result_type [inherited]
```

result\_type is the return type

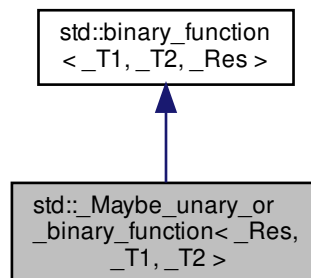
Definition at line 111 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

## 5.558 std::\_Maybe\_unary\_or\_binary\_function&lt;\_Res, \_T1, \_T2&gt; Struct Template Reference

Inheritance diagram for std::\_Maybe\_unary\_or\_binary\_function<\_Res, \_T1, \_T2>:



## Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `_Res` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### 5.558.1 Detailed Description

```
template<typename _Res, typename _T1, typename _T2>  
struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >
```

Derives from `binary_function`, as appropriate.

Definition at line 62 of file `refwrap.h`.

### 5.558.2 Member Typedef Documentation

#### 5.558.2.1 `first_argument_type`

```
typedef _T1 std::binary\_function< _T1 , _T2 , _Res >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.558.2.2 `result_type`

```
typedef _Res std::binary\_function< _T1 , _T2 , _Res >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 5.558.2.3 `second_argument_type`

```
typedef _T2 std::binary\_function< _T1 , _T2 , _Res >::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

## 5.559 `std::_Mu<_Arg, _IsBindExp, _IsPlaceholder >` Class Template Reference

### 5.559.1 Detailed Description

```
template<typename _Arg, bool _IsBindExp = is_bind_expression<_Arg>::value, bool _IsPlaceholder = (is_placeholder<_Arg>::value > 0)>
class std::_Mu<_Arg, _IsBindExp, _IsPlaceholder >
```

Maps an argument to `bind()` into an actual argument to the bound function object `[func.bind.bind]/10`. Only the first parameter should be specified: the rest are used to determine among the various implementations. Note that, although this class is a function object, it isn't entirely normal because it takes only two parameters regardless of the number of parameters passed to the bind expression. The first parameter is the bound argument and the second parameter is a tuple containing references to the rest of the arguments.

Definition at line 278 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.560 `std::_Mu<_Arg, false, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVarArg, typename _Tuple >`  
`_CVarArg && operator() (_CVarArg && __arg, _Tuple &) const volatile`

### 5.560.1 Detailed Description

```
template<typename _Arg>
class std::_Mu<_Arg, false, false >
```

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are determined by the caller. C++11 `[func.bind.bind]` p10 bullet 4.

Definition at line 358 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.561 `std::_Mu<_Arg, false, true >` Class Template Reference

### Public Member Functions

- `template<typename _Tuple >`  
`_Safe_tuple_element_t<(is_placeholder<_Arg>::value - 1), _Tuple > && operator() (const volatile _Arg &, _Tuple & __tuple) const volatile`

#### 5.561.1 Detailed Description

```
template<typename _Arg>
class std::_Mu< _Arg, false, true >
```

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. C++11 [func.bind.bind] p10 bullet 3.

Definition at line 340 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

#### 5.562 std::\_Mu< \_Arg, true, false > Class Template Reference

##### Public Member Functions

- `template<typename _CVArg, typename... _Args>`  
auto **operator()** (\_CVArg &\_\_arg, [tuple](#)< \_Args... > &\_\_tuple) const volatile -> decltype(\_\_arg(declval< \_Args >()...))

#### 5.562.1 Detailed Description

```
template<typename _Arg>
class std::_Mu< _Arg, true, false >
```

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). C++11 [func.bind.bind] p10 bullet 2.

Definition at line 306 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

#### 5.563 std::\_Mu< reference\_wrapper< \_Tp >, false, false > Class Template Reference

##### Public Member Functions

- `template<typename _CVRef, typename _Tuple >`  
\_Tp & **operator()** (\_CVRef &\_\_arg, \_Tuple &) const volatile

## 5.563.1 Detailed Description

```
template<typename _Tp>
class std::_Mu< reference_wrapper< _Tp >, false, false >
```

If the argument is reference\_wrapper<\_Tp>, returns the underlying reference. C++11 [func.bind.bind] p10 bullet 1.

Definition at line 286 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.564 std::\_Not\_fn&lt;\_Fn&gt; Class Template Reference

## Public Member Functions

- template<typename \_Fn2 >  
\_Not\_fn (\_Fn2 && \_\_fn, int)
- \_Not\_fn (const \_Not\_fn & \_\_fn)=default
- \_Not\_fn (\_Not\_fn && \_\_fn)=default

## Public Attributes

- template<typename... \_Args>  
decltype(\_S\_not< \_\_inv\_res\_t< \_Fn &, \_Args... >>()) **operator()** (\_Args &&... \_\_args) &&noexcept(\_\_is\_nothrow\_invocable< \_Fn &, \_Args... >::value &&noexcept(\_S\_not< \_\_inv\_res\_t< \_Fn &, \_Args... >>()))
- template<typename... \_Args>  
decltype(\_S\_not< \_\_inv\_res\_t< \_Fn const &, \_Args... >>()) **operator()** (\_Args &&... \_\_args) const &&noexcept(\_\_is\_nothrow\_invocable< \_Fn const &, \_Args... >::value &&noexcept(\_S\_not< \_\_inv\_res\_t< \_Fn const &, \_Args... >>()))
- template<typename... \_Args>  
decltype(\_S\_not< \_\_inv\_res\_t< \_Fn &&, \_Args... >>()) **operator()** (\_Args &&... \_\_args) &&noexcept(\_\_is\_nothrow\_invocable< \_Fn &&, \_Args... >::value &&noexcept(\_S\_not< \_\_inv\_res\_t< \_Fn &&, \_Args... >>()))
- template<typename... \_Args>  
decltype(\_S\_not< \_\_inv\_res\_t< \_Fn const &&, \_Args... >>()) **operator()** (\_Args &&... \_\_args) const &&noexcept(\_\_is\_nothrow\_invocable< \_Fn const &&, \_Args... >::value &&noexcept(\_S\_not< \_\_inv\_res\_t< \_Fn const &&, \_Args... >>()))

## 5.564.1 Detailed Description

```
template<typename _Fn>
class std::_Not_fn< _Fn >
```

Generalized negator.

Definition at line 842 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.565 `std::_Placeholder<_Num>` Struct Template Reference

### 5.565.1 Detailed Description

```
template<int _Num>
struct std::_Placeholder<_Num>
```

The type of placeholder objects defined by libstdc++.

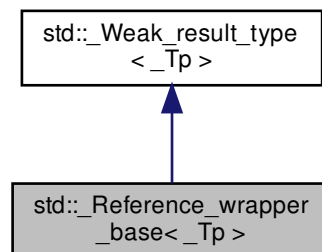
Definition at line 199 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.566 `std::_Reference_wrapper_base<_Tp>` Struct Template Reference

Inheritance diagram for `std::_Reference_wrapper_base<_Tp>`:



### 5.566.1 Detailed Description

```
template<typename _Tp>
struct std::_Reference_wrapper_base<_Tp>
```

Derives from `unary_function` or `binary_function` when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both `unary_function` and `binary_function`.

Definition at line 213 of file refwrap.h.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

## 5.567 std::\_Sp\_ebo\_helper< \_Nm, \_Tp, false > Struct Template Reference

### Public Member Functions

- **\_Sp\_ebo\_helper** (const \_Tp &\_\_tp)
- **\_Sp\_ebo\_helper** (\_Tp &&\_\_tp)

### Static Public Member Functions

- static \_Tp & **S\_get** (\_Sp\_ebo\_helper &\_\_eboh)

#### 5.567.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper< _Nm, _Tp, false >
```

Specialization not using EBO.

Definition at line 423 of file shared\_ptr\_base.h.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 5.568 std::\_Sp\_ebo\_helper< \_Nm, \_Tp, true > Struct Template Reference

Inherits \_Tp.

### Public Member Functions

- **\_Sp\_ebo\_helper** (const \_Tp &\_\_tp)
- **\_Sp\_ebo\_helper** (\_Tp &&\_\_tp)

### Static Public Member Functions

- static \_Tp & **S\_get** (\_Sp\_ebo\_helper &\_\_eboh)

#### 5.568.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper< _Nm, _Tp, true >
```

Specialization using EBO.

Definition at line 412 of file shared\_ptr\_base.h.

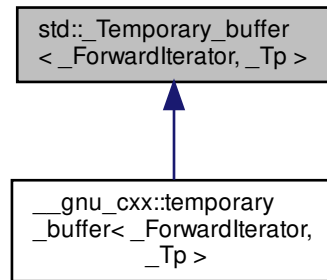
The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)



## 5.569 std::\_Temporary\_buffer<\_ForwardIterator, \_Tp> Class Template Reference

Inheritance diagram for std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- [\\_Temporary\\_buffer](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

### Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

### 5.569.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer<_ForwardIterator, _Tp>
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 122 of file `stl_tempbuf.h`.

## 5.569.2 Constructor & Destructor Documentation

### 5.569.2.1 \_Temporary\_buffer()

```
template<typename _ForwardIterator , typename _Tp >
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
    _ForwardIterator __first,
    _ForwardIterator __last )
```

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file stl\_tempbuf.h.

## 5.569.3 Member Function Documentation

### 5.569.3.1 begin()

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( ) [inline]
```

As per Table mumble.

Definition at line 151 of file stl\_tempbuf.h.

### 5.569.3.2 end()

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( ) [inline]
```

As per Table mumble.

Definition at line 156 of file stl\_tempbuf.h.

### 5.569.3.3 requested\_size()

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size ( ) const [inline]
```

Returns the size requested by the constructor; may be >size().

Definition at line 146 of file stl\_tempbuf.h.

#### 5.569.3.4 size()

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const [inline]
```

As per Table mumble.

Definition at line 141 of file `stl_tempbuf.h`.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

### 5.570 std::\_Tuple\_impl<\_Idx, \_Elements > Struct Template Reference

#### 5.570.1 Detailed Description

```
template<std::size_t _Idx, typename... _Elements>
struct std::_Tuple_impl< _Idx, _Elements >
```

Contains the actual implementation of the `tuple` template, stored as a recursive inheritance hierarchy from the first element (most derived class) to the last (least derived class). The `Idx` parameter gives the 0-based index of the element stored at this point in the hierarchy; we use it to implement a constant-time `get()` operation.

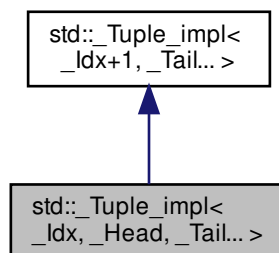
Definition at line 177 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

### 5.571 std::\_Tuple\_impl<\_Idx, \_Head, \_Tail... > Struct Template Reference

Inheritance diagram for `std::_Tuple_impl<_Idx, _Head, _Tail... >`:



## Public Types

- typedef `_Head_base`< `_Idx`, `_Head` > `_Base`
- typedef `_Tuple_impl`< `_Idx+1`, `_Tail...` > `_Inherited`

## Public Member Functions

- constexpr `_Tuple_impl` (const `_Head` & `__head`, const `_Tail` &... `__tail`)
- template<typename `_UHead` , typename... `_UTail`, typename = typename enable\_if<sizeof...(`_Tail`) == sizeof...(`_UTail`)>::type>  
constexpr `_Tuple_impl` (`_UHead` && `__head`, `_UTail` &&... `__tail`)
- constexpr `_Tuple_impl` (const `_Tuple_impl` &)=default
- constexpr `_Tuple_impl` (`_Tuple_impl` && `__in`) noexcept(\_\_and\_< `is_nothrow_move_constructible`< `_Head` >, `is_nothrow_move_constructible`< `_Inherited` >>::value)
- template<typename... `_UElements`>  
constexpr `_Tuple_impl` (const `_Tuple_impl`< `_Idx`, `_UElements...` > & `__in`)
- template<typename `_UHead` , typename... `_UTails`>  
constexpr `_Tuple_impl` (`_Tuple_impl`< `_Idx`, `_UHead`, `_UTails...` > && `__in`)
- template<typename `_Alloc` >  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`)
- template<typename `_Alloc` >  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`, const `_Head` & `__head`, const `_Tail` &... `__tail`)
- template<typename `_Alloc` , typename `_UHead` , typename... `_UTail`, typename = typename enable\_if<sizeof...(`_Tail`) == sizeof...(`_U`↔`_Tail`)>::type>  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`, `_UHead` && `__head`, `_UTail` &&... `__tail`)
- template<typename `_Alloc` >  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`, const `_Tuple_impl` & `__in`)
- template<typename `_Alloc` >  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`, `_Tuple_impl` && `__in`)
- template<typename `_Alloc` , typename `_UHead` , typename... `_UTails`>  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`, const `_Tuple_impl`< `_Idx`, `_UHead`, `_UTails...` > & `__in`)
- template<typename `_Alloc` , typename `_UHead` , typename... `_UTails`>  
`_Tuple_impl` (`allocator_arg_t` `__tag`, const `_Alloc` & `__a`, `_Tuple_impl`< `_Idx`, `_UHead`, `_UTails...` > && `__in`)
- `_Tuple_impl` & `operator=` (const `_Tuple_impl` & `__in`)
- `_Tuple_impl` & `operator=` (`_Tuple_impl` && `__in`) noexcept(\_\_and\_< `is_nothrow_move_assignable`< `_Head` >, `is_nothrow_move_assignable`< `_Inherited` >>::value)
- template<typename... `_UElements`>  
`_Tuple_impl` & `operator=` (const `_Tuple_impl`< `_Idx`, `_UElements...` > & `__in`)
- template<typename `_UHead` , typename... `_UTails`>  
`_Tuple_impl` & `operator=` (`_Tuple_impl`< `_Idx`, `_UHead`, `_UTails...` > && `__in`)

## Static Public Member Functions

- static constexpr `_Head` & `_M_head` (`_Tuple_impl` & `__t`) noexcept
- static constexpr const `_Head` & `_M_head` (const `_Tuple_impl` & `__t`) noexcept
- static constexpr `_Inherited` & `_M_tail` (`_Tuple_impl` & `__t`) noexcept
- static constexpr const `_Inherited` & `_M_tail` (const `_Tuple_impl` & `__t`) noexcept

## Protected Member Functions

- void `_M_swap` (`_Tuple_impl` & `__in`) noexcept(\_\_is\_nothrow\_swappable< `_Head` >::value &&noexcept(\_M\_↔`tail`(`__in`).\_M\_swap(\_M\_tail(`__in`))))

## Friends

- `template<std::size_t , typename... >`  
`class _Tuple_impl`

### 5.571.1 Detailed Description

```
template<std::size_t _Idx, typename _Head, typename... _Tail>
struct std::_Tuple_impl<_Idx, _Head, _Tail... >
```

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

Definition at line 185 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.572 std::\_V2::condition\_variable\_any Class Reference

### Public Member Functions

- **condition\_variable\_any** (const [condition\\_variable\\_any](#) &)=delete
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable\\_any](#) & **operator=** (const [condition\\_variable\\_any](#) &)=delete
- `template<typename _Lock >`  
void **wait** (\_Lock &\_\_lock)
- `template<typename _Lock , typename _Predicate >`  
void **wait** (\_Lock &\_\_lock, \_Predicate \_\_p)
- `template<typename _Lock , typename _Rep , typename _Period >`  
[cv\\_status](#) **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- `template<typename _Lock , typename _Rep , typename _Period , typename _Predicate >`  
bool **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- `template<typename _Lock , typename _Clock , typename _Duration >`  
[cv\\_status](#) **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- `template<typename _Lock , typename _Clock , typename _Duration , typename _Predicate >`  
bool **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

### 5.572.1 Detailed Description

`condition_variable_any`

Definition at line 199 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 5.573 std::\_V2::error\_category Class Reference

## Public Member Functions

- **error\_category** (const [error\\_category](#) &)=delete
- virtual [error\\_condition](#) **default\_error\_condition** (int \_\_i) const noexcept
- virtual bool **equivalent** (int \_\_i, const [error\\_condition](#) &\_\_cond) const noexcept
- virtual bool **equivalent** (const [error\\_code](#) &\_\_code, int \_\_i) const noexcept
- virtual [string](#) **message** (int) const =0
- virtual const char \* **name** () const noexcept=0
- bool **operator!=** (const [error\\_category](#) &\_\_other) const noexcept
- bool **operator<** (const [error\\_category](#) &\_\_other) const noexcept
- [error\\_category](#) & **operator=** (const [error\\_category](#) &)=delete
- bool **operator==** (const [error\\_category](#) &\_\_other) const noexcept

## 5.573.1 Detailed Description

error\_category

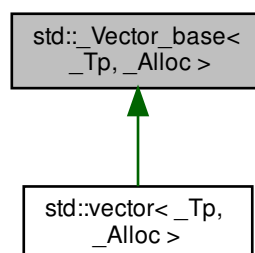
Definition at line 74 of file system\_error.

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.574 std::\_Vector\_base&lt; \_Tp, \_Alloc &gt; Struct Template Reference

Inheritance diagram for std::\_Vector\_base< \_Tp, \_Alloc >:



### Public Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)<\_Alloc >::template rebind<\_Tp >::other **\_Tp\_alloc\_type**
- typedef \_Alloc **allocator\_type**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)<\_Tp\_alloc\_type >::pointer **pointer**

### Public Member Functions

- **\_Vector\_base** (const allocator\_type &\_\_a) noexcept
- **\_Vector\_base** (size\_t \_\_n)
- **\_Vector\_base** (size\_t \_\_n, const allocator\_type &\_\_a)
- **\_Vector\_base** (\_Tp\_alloc\_type &&\_\_a) noexcept
- **\_Vector\_base** ([\\_Vector\\_base](#) &&\_\_x) noexcept
- **\_Vector\_base** ([\\_Vector\\_base](#) &&\_\_x, const allocator\_type &\_\_a)
- pointer **\_M\_allocate** (size\_t \_\_n)
- void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- allocator\_type **get\_allocator** () const noexcept

### Public Attributes

- [\\_Vector\\_impl](#) **\_M\_impl**

#### 5.574.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::_Vector_base<_Tp, _Alloc >
```

See bits/stl\_deque.h's `_Deque_base` for an explanation.

Definition at line 81 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

#### 5.575 `std::_Weak_result_type<_Functor >` Struct Template Reference

Inherits `std::_Weak_result_type_memfun<_Functor, bool >`.

Inherited by `std::_Bind<_Functor(_Bound_args...)>`.

## 5.575.1 Detailed Description

```
template<typename _Functor>
struct std::_Weak_result_type<_Functor>
```

Strip top-level cv-qualifiers from the function object and let \_Weak\_result\_type\_memfun perform the real work.

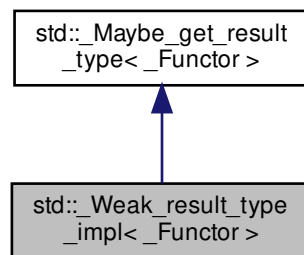
Definition at line 174 of file refwrap.h.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

## 5.576 std::\_Weak\_result\_type\_impl&lt;\_Functor&gt; Struct Template Reference

Inheritance diagram for std::\_Weak\_result\_type\_impl<\_Functor>:



## 5.576.1 Detailed Description

```
template<typename _Functor>
struct std::_Weak_result_type_impl<_Functor>
```

Base class for any function object that has a weak result type, as defined in 20.8.2 [func.require] of C++11.

Definition at line 125 of file refwrap.h.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)



### 5.577 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL >` Struct Template Reference

#### Public Types

- `typedef _Res result_type`

#### 5.577.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes _GLIBCXX_NOEXCEPT_PARM>
struct std::_Weak_result_type_impl< _Res(*)(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL >
```

Retrieve the result type for a function pointer.

Definition at line 141 of file `refwrap.h`.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

### 5.578 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL >` Struct Template Reference

#### Public Types

- `typedef _Res result_type`

#### 5.578.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes _GLIBCXX_NOEXCEPT_PARM>
struct std::_Weak_result_type_impl< _Res(*)(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL >
```

Retrieve the result type for a varargs function pointer.

Definition at line 146 of file `refwrap.h`.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

### 5.579 `std::_Weak_result_type_impl< _Res(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL >` Struct Template Reference

#### Public Types

- `typedef _Res result_type`

#### 5.579.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes _GLIBCXX_NOEXCEPT_PARM>
struct std::_Weak_result_type_impl< _Res(_ArgTypes...) _GLIBCXX_NOEXCEPT_QUAL >
```

Retrieve the result type for a function type.

Definition at line 131 of file `refwrap.h`.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

### 5.580 `std::_Weak_result_type_impl< _Res(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL >` Struct Template Reference

#### Public Types

- typedef `_Res` **result\_type**

#### 5.580.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes _GLIBCXX_NOEXCEPT_PARM>
struct std::_Weak_result_type_impl< _Res(_ArgTypes.....) _GLIBCXX_NOEXCEPT_QUAL >
```

Retrieve the result type for a varargs function type.

Definition at line 136 of file `refwrap.h`.

The documentation for this struct was generated from the following file:

- [refwrap.h](#)

### 5.581 `std::add_const< _Tp >` Struct Template Reference

#### Public Types

- typedef `_Tp` const **type**

#### 5.581.1 Detailed Description

```
template<typename _Tp>  
struct std::add_const<_Tp >
```

add\_const

Definition at line 1355 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.582 std::add\_cv<\_Tp > Struct Template Reference

##### Public Types

- typedef [add\\_const](#)< typename [add\\_volatile](#)<\_Tp >::type >::type **type**

#### 5.582.1 Detailed Description

```
template<typename _Tp>  
struct std::add_cv<_Tp >
```

add\_cv

Definition at line 1365 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.583 std::add\_lvalue\_reference<\_Tp > Struct Template Reference

Inherits std::\_\_add\_lvalue\_reference\_helper<\_Tp, bool >.

##### Public Types

- typedef \_Tp **type**

## 5.583.1 Detailed Description

```
template<typename _Tp>  
struct std::add_lvalue_reference< _Tp >
```

add\_lvalue\_reference

Definition at line 1425 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.584 std::add\_rvalue\_reference&lt; \_Tp &gt; Struct Template Reference

Inherits std::\_\_add\_rvalue\_reference\_helper< \_Tp, bool >.

## Public Types

- typedef \_Tp **type**

## 5.584.1 Detailed Description

```
template<typename _Tp>  
struct std::add_rvalue_reference< _Tp >
```

add\_rvalue\_reference

Definition at line 1439 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.585 std::add\_volatile&lt; \_Tp &gt; Struct Template Reference

## Public Types

- typedef \_Tp volatile **type**

#### 5.585.1 Detailed Description

```
template<typename _Tp>
struct std::add_volatile< _Tp >
```

add\_volatile

Definition at line 1360 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.586 std::adopt\_lock\_t Struct Reference

##### 5.586.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.

Definition at line 139 of file std\_mutex.h.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

#### 5.587 std::aligned\_storage< \_Len, \_Align > Struct Template Reference

##### 5.587.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>
struct std::aligned_storage< _Len, _Align >
```

Alignment type.

The value of \_Align is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than \_Len (3.9). The member typedef type shall be a POD type suitable for use as uninitialized storage for any object whose size is at most \_Len and whose alignment is a divisor of \_Align.

Definition at line 1793 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.588 `std::aligned_union<_Len, _Types >` Struct Template Reference

### Public Types

- typedef `aligned_storage<_S_len, alignment_value >::type` `type`

### Static Public Attributes

- static const size\_t `alignment_value`

### 5.588.1 Detailed Description

```
template<size_t _Len, typename... _Types>
struct std::aligned_union<_Len, _Types >
```

Provide aligned storage for types.

[meta.trans.other]

Provides aligned storage for any of the provided types of at least size `_Len`.

### See also

`aligned_storage`

Definition at line 1831 of file `type_traits`.

### 5.588.2 Member Typedef Documentation

#### 5.588.2.1 `type`

```
template<size_t _Len, typename... _Types>
typedef aligned_storage<_S_len, alignment_value>::type std::aligned_union<_Len, _Types >::type
```

The storage.

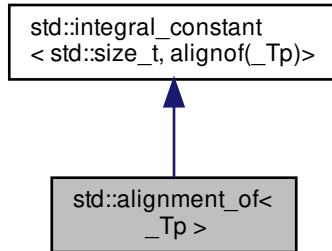
Definition at line 1843 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

## 5.589 std::alignment\_of< \_Tp > Struct Template Reference

Inheritance diagram for std::alignment\_of< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr std::size\_t **value**

### 5.589.1 Detailed Description

```
template<typename _Tp>
struct std::alignment_of< _Tp >
```

alignment\_of

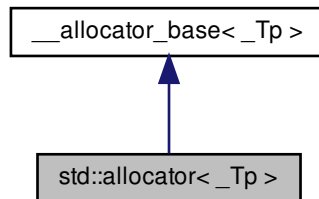
Definition at line 1239 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.590 std::allocator&lt; \_Tp &gt; Class Template Reference

Inheritance diagram for std::allocator< \_Tp >:



## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef [true\\_type](#) **is\_always\_equal**
- typedef \_Tp \* **pointer**
- typedef [true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **allocator** (const [allocator](#) &\_\_a) throw ()
- template<typename \_Tp1 >  
**allocator** (const [allocator](#)< \_Tp1 > &) throw ()
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \* \_\_p) const noexcept
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up >  
void **destroy** (\_Up \* \_\_p)
- size\_type **max\_size** () const noexcept



#### 5.590.1 Detailed Description

```
template<typename _Tp>  
class std::allocator<_Tp>
```

The *standard* allocator, as per [20.4].

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.↵  
memory.allocator](https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator) for further details.

## Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 108 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

5.591 `std::allocator< void >` Class Template Reference

## Public Types

- typedef const void \* **const\_pointer**
- typedef ptrdiff\_t **difference\_type**
- typedef [true\\_type](#) **is\_always\_equal**
- typedef void \* **pointer**
- typedef [true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef void **value\_type**

## Public Member Functions

- template<typename `_Up` , typename... `_Args`>  
void **construct** (`_Up` \*`__p`, `_Args` &&... `__args`)
- template<typename `_Up` >  
void **destroy** (`_Up` \*`__p`)

## 5.591.1 Detailed Description

```
template<>
class std::allocator< void >
```

`allocator<void>` specialization.

Definition at line 68 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 5.592 std::allocator\_arg\_t Struct Reference

### 5.592.1 Detailed Description

[allocator.tag]

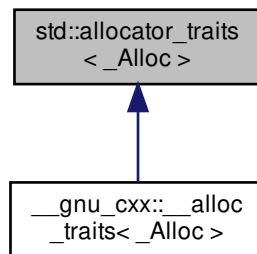
Definition at line 46 of file uses\_allocator.h.

The documentation for this struct was generated from the following file:

- uses\_allocator.h

## 5.593 std::allocator\_traits<\_Alloc> Struct Template Reference

Inheritance diagram for std::allocator\_traits<\_Alloc>:



### Public Types

- typedef `_Alloc` `allocator_type`
- using `const_pointer` = typename `_Ptr`< `__c_pointer`, const `value_type` >::type
- using `const_void_pointer` = typename `_Ptr`< `__cv_pointer`, const void >::type
- using `difference_type` = typename `_Diff`< `_Alloc`, `pointer` >::type
- using `is_always_equal` = `__detected_or_t`< typename `is_empty`< `_Alloc` >::type, `__equal`, `_Alloc` >
- using `pointer` = `__detected_or_t`< `value_type` \*, `__pointer`, `_Alloc` >
- using `propagate_on_container_copy_assignment` = `__detected_or_t`< `false_type`, `__pocca`, `_Alloc` >
- using `propagate_on_container_move_assignment` = `__detected_or_t`< `false_type`, `__pocma`, `_Alloc` >
- using `propagate_on_container_swap` = `__detected_or_t`< `false_type`, `__pocs`, `_Alloc` >
- template<typename `_Tp` >  
using `rebind_alloc` = `__alloc_rebind`< `_Alloc`, `_Tp` >
- template<typename `_Tp` >  
using `rebind_traits` = `allocator_traits`< `rebind_alloc`< `_Tp` > >
- using `size_type` = typename `_Size`< `_Alloc`, `difference_type` >::type
- typedef `_Alloc::value_type` `value_type`
- using `void_pointer` = typename `_Ptr`< `__v_pointer`, void >::type

## Static Public Member Functions

- static [pointer allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n)
- static [pointer allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- template<typename \_Tp, typename... \_Args>  
static auto [construct](#) (\_Alloc &\_\_a, \_Tp \*\_\_p, \_Args &&... \_\_args) -> decltype(\_S\_construct(\_\_a, \_\_p, [std::forward](#)<\_Args>(\_\_args)...))
- static void [deallocate](#) (\_Alloc &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- template<typename \_Tp>  
static void [destroy](#) (\_Alloc &\_\_a, \_Tp \*\_\_p)
- static [size\\_type max\\_size](#) (const \_Alloc &\_\_a) noexcept
- static \_Alloc [select\\_on\\_container\\_copy\\_construction](#) (const \_Alloc &\_\_rhs)

## Protected Types

- template<typename \_Tp>  
using [\\_\\_c\\_pointer](#) = typename \_Tp::const\_pointer
- template<typename \_Tp>  
using [\\_\\_cv\\_pointer](#) = typename \_Tp::const\_void\_pointer
- template<typename \_Tp>  
using [\\_\\_equal](#) = typename \_Tp::is\_always\_equal
- template<typename \_Tp>  
using [\\_\\_pocca](#) = typename \_Tp::propagate\_on\_container\_copy\_assignment
- template<typename \_Tp>  
using [\\_\\_pocma](#) = typename \_Tp::propagate\_on\_container\_move\_assignment
- template<typename \_Tp>  
using [\\_\\_pocs](#) = typename \_Tp::propagate\_on\_container\_swap
- template<typename \_Tp>  
using [\\_\\_pointer](#) = typename \_Tp::pointer
- template<typename \_Tp>  
using [\\_\\_v\\_pointer](#) = typename \_Tp::void\_pointer

## 5.593.1 Detailed Description

```
template<typename _Alloc>
struct std::allocator_traits<_Alloc>
```

Uniform interface to all allocator types.

Definition at line 83 of file bits/alloc\_traits.h.

## 5.593.2 Member Typedef Documentation

#### 5.593.2.1 allocator\_type

```
template<typename _Alloc>
typedef _Alloc std::allocator_traits< _Alloc >::allocator_type
```

The allocator type.

Definition at line 86 of file bits/alloc\_traits.h.

#### 5.593.2.2 const\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_pointer = typename _Ptr<__c_pointer, const value_type>↵
::type
```

The allocator's const pointer type.

`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

Definition at line 135 of file bits/alloc\_traits.h.

#### 5.593.2.3 const\_void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer = typename _Ptr<__cv_pointer, const
void>::type
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 151 of file bits/alloc\_traits.h.

#### 5.593.2.4 difference\_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::difference_type = typename _Diff<_Alloc, pointer>::type
```

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference↵_type`

Definition at line 159 of file bits/alloc\_traits.h.

#### 5.593.2.5 is\_always\_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal = __detected_or_t<typename is_empty<_Alloc>::type, __equal, _Alloc>
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

Definition at line 203 of file bits/alloc\_traits.h.

#### 5.593.2.6 pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::pointer = __detected_or_t<value_type*, __pointer, _Alloc>
```

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

Definition at line 95 of file bits/alloc\_traits.h.

#### 5.593.2.7 propagate\_on\_container\_copy\_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment = __detected_or_t<false_type, __pocca, _Alloc>
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 176 of file bits/alloc\_traits.h.

#### 5.593.2.8 propagate\_on\_container\_move\_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment = __detected_or_t<false_type, __pocma, _Alloc>
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 185 of file bits/alloc\_traits.h.

### 5.593.2.9 propagate\_on\_container\_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap = __detected_or_t<false_type,
__pocs, _Alloc>
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 194 of file `bits/alloc_traits.h`.

### 5.593.2.10 size\_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::size_type = typename _Size<_Alloc, difference_type>::type
```

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

Definition at line 167 of file `bits/alloc_traits.h`.

### 5.593.2.11 value\_type

```
template<typename _Alloc>
typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type
```

The allocated type.

Definition at line 88 of file `bits/alloc_traits.h`.

### 5.593.2.12 void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer = typename _Ptr<__v_pointer, void>::type
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 143 of file `bits/alloc_traits.h`.

## 5.593.3 Member Function Documentation

### 5.593.3.1 allocate() [1/2]

```
template<typename _Alloc>
static pointer std::allocator_traits< _Alloc >::allocate (
    _Alloc & __a,
    size_type __n ) [inline], [static]
```

Allocate memory.

## Parameters

<code>__a</code>	An allocator.
<code>n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 300 of file `bits/alloc_traits.h`.

Referenced by `std::__allocate_guarded()`.

## 5.593.3.2 allocate() [2/2]

```
template<typename _Alloc>
static pointer std::allocator_traits<_Alloc>::allocate (
    _Alloc & __a,
    size_type __n,
    const_void_pointer __hint ) [inline], [static]
```

Allocate memory.

## Parameters

<code>__a</code>	An allocator.
<code>n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

## Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 315 of file `bits/alloc_traits.h`.

## 5.593.3.3 construct()

```
template<typename _Alloc>
template<typename _Tp, typename... _Args>
static auto std::allocator_traits<_Alloc>::construct (
    _Alloc & __a,
    _Tp * __p,
    _Args &&... __args ) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...) ) [inline], [static]
```

Construct an object of type `_Tp`.



## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

Definition at line 342 of file `bits/alloc_traits.h`.

5.593.3.4 `deallocate()`

```
template<typename _Alloc>
static void std::allocator_traits< _Alloc >::deallocate (
    _Alloc & __a,
    pointer __p,
    size_type __n ) [inline], [static]
```

Deallocate memory.

## Parameters

<code>__↵ __a</code>	An allocator.
<code>__↵ __p</code>	Pointer to the memory to deallocate.
<code>__↵ __n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 327 of file `bits/alloc_traits.h`.

Referenced by `std::__allocated_ptr< _Alloc >::~~__allocated_ptr()`.

5.593.3.5 `destroy()`

```
template<typename _Alloc>
template<typename _Tp >
static void std::allocator_traits< _Alloc >::destroy (
    _Alloc & __a,
    _Tp * __p ) [inline], [static]
```

Destroy an object of type `_Tp`.

## Parameters

$\leftrightarrow$ _a	An allocator.
$\leftrightarrow$ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~Tp()`

Definition at line 355 of file `bits/alloc_traits.h`.

## 5.593.3.6 max\_size()

```
template<typename _Alloc>
static size_type std::allocator_traits<_Alloc>::max_size (
    const _Alloc & __a ) [inline], [static], [noexcept]
```

The maximum supported allocation size.

## Parameters

$\leftrightarrow$ _a	An allocator.
-------------------------	---------------

## Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 366 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`, and `std::list<__inp, __rebind_inp>::max_size()`.

## 5.593.3.7 select\_on\_container\_copy\_construction()

```
template<typename _Alloc>
static _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction (
    const _Alloc & __rhs ) [inline], [static]
```

Obtain an allocator to use when copying a container.

## Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 378 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

## 5.594 `std::allocator_traits< allocator< _Tp > >` Struct Template Reference

## Public Types

- using `allocator_type` = `allocator< _Tp >`
- using `const_pointer` = `const _Tp *`
- using `const_void_pointer` = `const void *`
- using `difference_type` = `std::ptrdiff_t`
- using `is_always_equal` = `true_type`
- using `pointer` = `_Tp *`
- using `propagate_on_container_copy_assignment` = `false_type`
- using `propagate_on_container_move_assignment` = `true_type`
- using `propagate_on_container_swap` = `false_type`
- template<typename `_Up` >  
using `rebind_alloc` = `allocator< _Up >`
- template<typename `_Up` >  
using `rebind_traits` = `allocator_traits< allocator< _Up > >`
- using `size_type` = `std::size_t`
- using `value_type` = `_Tp`
- using `void_pointer` = `void *`

## Static Public Member Functions

- static `pointer allocate` (`allocator_type` &`__a`, `size_type` `__n`)
- static `pointer allocate` (`allocator_type` &`__a`, `size_type` `__n`, `const_void_pointer` `__hint`)
- template<typename `_Up`, typename... `_Args`>  
static void `construct` (`allocator_type` &`__a`, `_Up *``__p`, `_Args` &&... `__args`)
- static void `deallocate` (`allocator_type` &`__a`, `pointer` `__p`, `size_type` `__n`)
- template<typename `_Up` >  
static void `destroy` (`allocator_type` &`__a`, `_Up *``__p`)
- static `size_type max_size` (const `allocator_type` &`__a`) noexcept
- static `allocator_type select_on_container_copy_construction` (const `allocator_type` &`__rhs`)

#### 5.594.1 Detailed Description

```
template<typename _Tp>  
struct std::allocator_traits< allocator< _Tp > >
```

Partial specialization for `std::allocator`.

Definition at line 384 of file `bits/alloc_traits.h`.

#### 5.594.2 Member Typedef Documentation

##### 5.594.2.1 `allocator_type`

```
template<typename _Tp >  
using std::allocator_traits< allocator< _Tp > >::allocator_type = allocator<_Tp>
```

The allocator type.

Definition at line 387 of file `bits/alloc_traits.h`.

##### 5.594.2.2 `const_pointer`

```
template<typename _Tp >  
using std::allocator_traits< allocator< _Tp > >::const_pointer = const _Tp*
```

The allocator's const pointer type.

Definition at line 395 of file `bits/alloc_traits.h`.

##### 5.594.2.3 `const_void_pointer`

```
template<typename _Tp >  
using std::allocator_traits< allocator< _Tp > >::const_void_pointer = const void*
```

The allocator's const void pointer type.

Definition at line 401 of file `bits/alloc_traits.h`.

#### 5.594.2.4 difference\_type

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::difference_type = std::ptrdiff_t
```

The allocator's difference type.

Definition at line 404 of file bits/alloc\_traits.h.

#### 5.594.2.5 is\_always\_equal

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::is_always_equal = true_type
```

Whether all instances of the allocator type compare equal.

Definition at line 419 of file bits/alloc\_traits.h.

#### 5.594.2.6 pointer

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::pointer = _Tp*
```

The allocator's pointer type.

Definition at line 392 of file bits/alloc\_traits.h.

#### 5.594.2.7 propagate\_on\_container\_copy\_assignment

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment = false_type
```

How the allocator is propagated on copy assignment.

Definition at line 410 of file bits/alloc\_traits.h.

#### 5.594.2.8 propagate\_on\_container\_move\_assignment

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment = true_type
```

How the allocator is propagated on move assignment.

Definition at line 413 of file bits/alloc\_traits.h.

#### 5.594.2.9 `propagate_on_container_swap`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap = false_type
```

How the allocator is propagated on swap.

Definition at line 416 of file `bits/alloc_traits.h`.

#### 5.594.2.10 `size_type`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::size_type = std::size_t
```

The allocator's size type.

Definition at line 407 of file `bits/alloc_traits.h`.

#### 5.594.2.11 `value_type`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::value_type = _Tp
```

The allocated type.

Definition at line 389 of file `bits/alloc_traits.h`.

#### 5.594.2.12 `void_pointer`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::void_pointer = void*
```

The allocator's void pointer type.

Definition at line 398 of file `bits/alloc_traits.h`.

### 5.594.3 Member Function Documentation

#### 5.594.3.1 `allocate()` [1/2]

```
template<typename _Tp >
static pointer std::allocator_traits< allocator< _Tp > >::allocate (
    allocator_type & __a,
    size_type __n ) [inline], [static]
```

Allocate memory.

**Parameters**

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 435 of file `bits/alloc_traits.h`.

**5.594.3.2 allocate()** [2/2]

```
template<typename _Tp >
static pointer std::allocator_traits< allocator< _Tp > >::allocate (
    allocator_type & __a,
    size_type __n,
    const_void_pointer __hint ) [inline], [static]
```

Allocate memory.

**Parameters**

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

Definition at line 449 of file `bits/alloc_traits.h`.

**5.594.3.3 construct()**

```
template<typename _Tp >
template<typename _Up , typename... _Args>
static void std::allocator_traits< allocator< _Tp > >::construct (
    allocator_type & __a,
    _Up * __p,
    _Args &&... __args ) [inline], [static]
```

Construct an object of type `_Up`.

## Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct (__p, std::forward<Args> (__args) ...)`

Definition at line 474 of file `bits/alloc_traits.h`.

## 5.594.3.4 deallocate()

```
template<typename _Tp >
static void std::allocator_traits< allocator< _Tp > >::deallocate (
    allocator_type & __a,
    pointer __p,
    size_type __n ) [inline], [static]
```

Deallocate memory.

## Parameters

<code>↔ __a</code>	An allocator.
<code>↔ __p</code>	Pointer to the memory to deallocate.
<code>↔ __n</code>	The number of objects space was allocated for.

Calls `a.deallocate (p, n)`

Definition at line 461 of file `bits/alloc_traits.h`.

## 5.594.3.5 destroy()

```
template<typename _Tp >
template<typename _Up >
static void std::allocator_traits< allocator< _Tp > >::destroy (
    allocator_type & __a,
    _Up * __p ) [inline], [static]
```

Destroy an object of type `_Up`.



**Parameters**

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)`.

Definition at line 486 of file `bits/alloc_traits.h`.

**5.594.3.6 `max_size()`**

```
template<typename _Tp >
static size_type std::allocator_traits< allocator< _Tp > >::max_size (
    const allocator_type & __a ) [inline], [static], [noexcept]
```

The maximum supported allocation size.

**Parameters**

<code>__a</code>	An allocator.
------------------	---------------

**Returns**

`__a.max_size()`

Definition at line 495 of file `bits/alloc_traits.h`.

**5.594.3.7 `select_on_container_copy_construction()`**

```
template<typename _Tp >
static allocator_type std::allocator_traits< allocator< _Tp > >::select_on_container_copy_↵
construction (
    const allocator_type & __rhs ) [inline], [static]
```

Obtain an allocator to use when copying a container.

**Parameters**

<code>__rhs</code>	An allocator.
--------------------	---------------

## Returns

`__rhs`

Definition at line 504 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

5.595 `std::array<_Tp, _Nm >` Struct Template Reference

## Public Types

- `typedef ::__array_traits<_Tp, _Nm > _AT_Type`
- `typedef const value_type * const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef std::ptrdiff_t difference_type`
- `typedef value_type * iterator`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `_GLIBCXX17_CONSTEXPR reference at (size_type __n)`
- `constexpr const_reference at (size_type __n) const`
- `_GLIBCXX17_CONSTEXPR reference back () noexcept`
- `constexpr const_reference back () const noexcept`
- `_GLIBCXX17_CONSTEXPR iterator begin () noexcept`
- `_GLIBCXX17_CONSTEXPR const_iterator begin () const noexcept`
- `_GLIBCXX17_CONSTEXPR const_iterator cbegin () const noexcept`
- `_GLIBCXX17_CONSTEXPR const_iterator cend () const noexcept`
- `_GLIBCXX17_CONSTEXPR const\_reverse\_iterator crbegin () const noexcept`
- `_GLIBCXX17_CONSTEXPR const\_reverse\_iterator crend () const noexcept`
- `_GLIBCXX17_CONSTEXPR pointer data () noexcept`
- `_GLIBCXX17_CONSTEXPR const_pointer data () const noexcept`
- `constexpr bool empty () const noexcept`
- `_GLIBCXX17_CONSTEXPR iterator end () noexcept`
- `_GLIBCXX17_CONSTEXPR const_iterator end () const noexcept`
- `void fill (const value_type & __u)`
- `_GLIBCXX17_CONSTEXPR reference front () noexcept`
- `constexpr const_reference front () const noexcept`
- `constexpr size_type max_size () const noexcept`
- `_GLIBCXX17_CONSTEXPR reference operator[] (size_type __n) noexcept`
- `constexpr const_reference operator[] (size_type __n) const noexcept`
- `_GLIBCXX17_CONSTEXPR reverse\_iterator rbegin () noexcept`
- `_GLIBCXX17_CONSTEXPR const\_reverse\_iterator rbegin () const noexcept`
- `_GLIBCXX17_CONSTEXPR reverse\_iterator rend () noexcept`
- `_GLIBCXX17_CONSTEXPR const\_reverse\_iterator rend () const noexcept`
- `constexpr size_type size () const noexcept`
- `void swap (array & __other) noexcept(_AT_Type::is_nothrow_swappable::value)`

## Public Attributes

- `_AT_Type::_Type _M_elems`

## 5.595.1 Detailed Description

```
template<typename _Tp, std::size_t _Nm>
struct std::array<_Tp, _Nm >
```

A standard container for storing a fixed size sequence of elements.

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

## Template Parameters

<i> Tp </i>	Type of element. Required to be a complete type.
<i> N </i>	Number of elements.

Definition at line 94 of file `array`.

The documentation for this struct was generated from the following file:

- [array](#)

5.596 `std::atomic<_Tp>` Struct Template Reference

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_Tp \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_Tp **exchange** (\_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_Tp **exchange** (\_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept

- `_Tp load (memory_order __m=memory_order_seq_cst) const noexcept`
- `_Tp load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator _Tp () const noexcept`
- `operator _Tp () const volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `_Tp operator= (_Tp __i) noexcept`
- `_Tp operator= (_Tp __i) volatile noexcept`
- `void store (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 5.596.1 Detailed Description

```
template<typename _Tp>
struct std::atomic<_Tp>
```

Generic atomic type, primary class template.

#### Template Parameters

<code>_Tp</code>	Type to be made atomic, must be trivially copyable.
------------------	---

Definition at line 58 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.597 std::atomic<\_Tp\*> Struct Template Reference

#### Public Types

- `typedef __atomic_base<_Tp*> __base_type`
- `typedef _Tp* __pointer_type`

#### Public Member Functions

- `atomic (const atomic &)=delete`
- `constexpr atomic (__pointer_type __p) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`

- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const volatile noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator __pointer_type () const noexcept`
- `operator __pointer_type () const volatile noexcept`
- `__pointer_type operator++ (int) noexcept`
- `__pointer_type operator++ (int) volatile noexcept`
- `__pointer_type operator++ () noexcept`
- `__pointer_type operator++ () volatile noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator-- (int) noexcept`
- `__pointer_type operator-- (int) volatile noexcept`
- `__pointer_type operator-- () noexcept`
- `__pointer_type operator-- () volatile noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__pointer_type operator= (__pointer_type __p) noexcept`
- `__pointer_type operator= (__pointer_type __p) volatile noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### Public Attributes

- `__base_type _M_b`

## 5.597.1 Detailed Description

```
template<typename _Tp>
struct std::atomic< _Tp * >
```

Partial specialization for pointer types.

Definition at line 352 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.598 `std::atomic< bool >` Struct Template Reference

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (bool \_\_i) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- bool **load** ([memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) const noexcept
- bool **load** ([memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) const volatile noexcept
- **operator bool** () const noexcept
- **operator bool** () const volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- bool **operator=** (bool \_\_i) noexcept
- bool **operator=** (bool \_\_i) volatile noexcept
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) noexcept
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile noexcept

### 5.598.1 Detailed Description

```
template<>
struct std::atomic< bool >
```

atomic<bool>

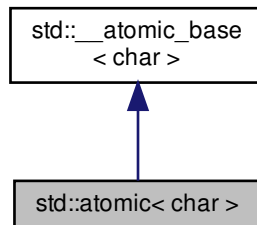
Definition at line 63 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.599 std::atomic< char > Struct Template Reference

Inheritance diagram for std::atomic< char >:



#### Public Types

- typedef [\\_\\_atomic\\_base](#)< char > **\_\_base\_type**
- typedef char **\_\_integral\_type**

#### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator&= (__int_type __i) volatile noexcept`
- `__int_type operator++ (int) noexcept`
- `__int_type operator++ (int) volatile noexcept`
- `__int_type operator++ () noexcept`
- `__int_type operator++ () volatile noexcept`
- `__int_type operator+= (__int_type __i) noexcept`
- `__int_type operator+= (__int_type __i) volatile noexcept`
- `__int_type operator-- (int) noexcept`
- `__int_type operator-- (int) volatile noexcept`
- `__int_type operator-- () noexcept`
- `__int_type operator-- () volatile noexcept`
- `__int_type operator-= (__int_type __i) noexcept`
- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`

#### 5.599.1 Detailed Description

```
template<>
struct std::atomic< char >
```

Explicit specialization for char.

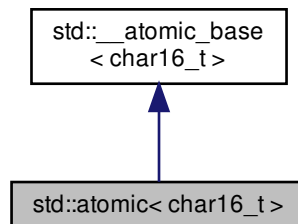
Definition at line 546 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 5.600 std::atomic< char16\_t > Struct Template Reference

Inheritance diagram for std::atomic< char16\_t >:





## Public Types

- typedef [\\_\\_atomic\\_base](#)< char16\_t > [\\_\\_base\\_type](#)
- typedef char16\_t [\\_\\_integral\\_type](#)

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i)
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

## 5.600.1 Detailed Description

```
template<>
struct std::atomic< char16_t >
```

Explicit specialization for char16\_t.

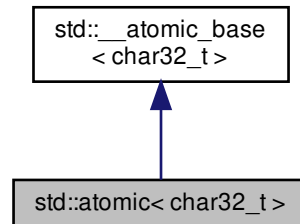
Definition at line 822 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.601 std::atomic&lt; char32\_t &gt; Struct Template Reference

Inheritance diagram for std::atomic< char32\_t >:



## Public Types

- typedef [\\_\\_atomic\\_base](#)< char32\_t > **\_\_base\_type**
- typedef char32\_t **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

### 5.601.1 Detailed Description

```
template<>
struct std::atomic< char32_t >
```

Explicit specialization for char32\_t.

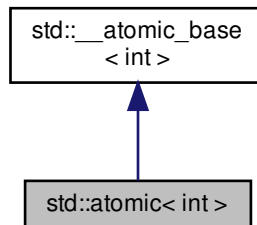
Definition at line 845 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.602 std::atomic< int > Struct Template Reference

Inheritance diagram for std::atomic< int >:



#### Public Types

- typedef [\\_\\_atomic\\_base](#)< int > **\_\_base\_type**
- typedef int **\_\_integral\_type**

#### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator&= (__int_type __i) volatile noexcept`
- `__int_type operator++ (int) noexcept`
- `__int_type operator++ (int) volatile noexcept`
- `__int_type operator++ () noexcept`
- `__int_type operator++ () volatile noexcept`
- `__int_type operator+= (__int_type __i) noexcept`
- `__int_type operator+= (__int_type __i) volatile noexcept`
- `__int_type operator-- (int) noexcept`
- `__int_type operator-- (int) volatile noexcept`
- `__int_type operator-- () noexcept`
- `__int_type operator-- () volatile noexcept`
- `__int_type operator-= (__int_type __i) noexcept`
- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`

#### 5.602.1 Detailed Description

```
template<>
struct std::atomic< int >
```

Explicit specialization for int.

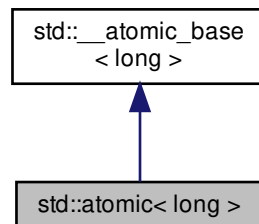
Definition at line 661 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.603 std::atomic< long > Struct Template Reference

Inheritance diagram for std::atomic< long >:



## Public Types

- typedef [\\_\\_atomic\\_base](#)< long > **\_\_base\_type**
- typedef long **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i)
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

## 5.603.1 Detailed Description

```
template<>
struct std::atomic< long >
```

Explicit specialization for long.

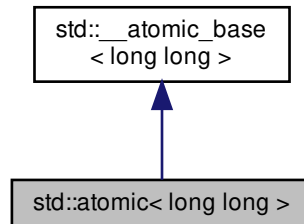
Definition at line 707 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.604 std::atomic&lt; long long &gt; Struct Template Reference

Inheritance diagram for std::atomic< long long >:



## Public Types

- typedef [\\_\\_atomic\\_base](#)< long long > **\_\_base\_type**
- typedef long long **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

#### 5.604.1 Detailed Description

```
template<>
struct std::atomic< long long >
```

Explicit specialization for long long.

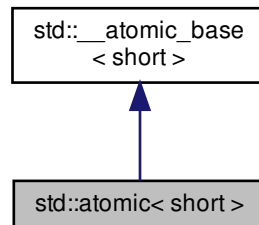
Definition at line 753 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 5.605 std::atomic< short > Struct Template Reference

Inheritance diagram for std::atomic< short >:



#### Public Types

- typedef [\\_\\_atomic\\_base](#)< short > **\_\_base\_type**
- typedef short **\_\_integral\_type**

#### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept

- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

#### 5.605.1 Detailed Description

template<>

struct std::atomic< short >

Explicit specialization for short.

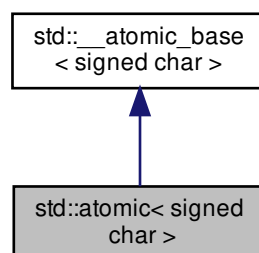
Definition at line 615 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.606 std::atomic< signed char > Struct Template Reference

Inheritance diagram for std::atomic< signed char >:





## Public Types

- typedef [\\_\\_atomic\\_base](#)< signed char > [\\_\\_base\\_type](#)
- typedef signed char [\\_\\_integral\\_type](#)

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i)
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

### 5.606.1 Detailed Description

```
template<>
struct std::atomic< signed char >
```

Explicit specialization for signed char.

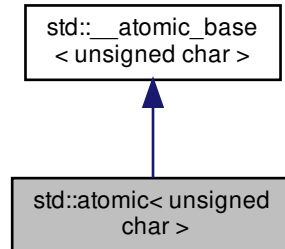
Definition at line 569 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.607 std::atomic&lt; unsigned char &gt; Struct Template Reference

Inheritance diagram for std::atomic< unsigned char >:



## Public Types

- typedef `__atomic_base< unsigned char > __base_type`
- typedef unsigned char `__integral_type`

## Public Member Functions

- **atomic** (const `atomic` &)=delete
- constexpr **atomic** (`__integral_type` \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(`__int_type` \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- `__int_type` **operator&=** (`__int_type` \_\_i) noexcept
- `__int_type` **operator&=** (`__int_type` \_\_i) volatile noexcept
- `__int_type` **operator++** (int) noexcept
- `__int_type` **operator++** (int) volatile noexcept
- `__int_type` **operator++** () noexcept
- `__int_type` **operator++** () volatile noexcept
- `__int_type` **operator+=** (`__int_type` \_\_i) noexcept
- `__int_type` **operator+=** (`__int_type` \_\_i) volatile noexcept
- `__int_type` **operator--** (int) noexcept
- `__int_type` **operator--** (int) volatile noexcept
- `__int_type` **operator--** () noexcept
- `__int_type` **operator--** () volatile noexcept
- `__int_type` **operator-=** (`__int_type` \_\_i) noexcept
- `__int_type` **operator-=** (`__int_type` \_\_i) volatile noexcept
- `atomic` & **operator=** (const `atomic` &)=delete
- `atomic` & **operator=** (const `atomic` &) volatile=delete
- `__int_type` **operator^=** (`__int_type` \_\_i) noexcept
- `__int_type` **operator^=** (`__int_type` \_\_i) volatile noexcept
- `__int_type` **operator|=** (`__int_type` \_\_i) noexcept
- `__int_type` **operator|=** (`__int_type` \_\_i) volatile noexcept

### 5.607.1 Detailed Description

```
template<>
struct std::atomic< unsigned char >
```

Explicit specialization for unsigned char.

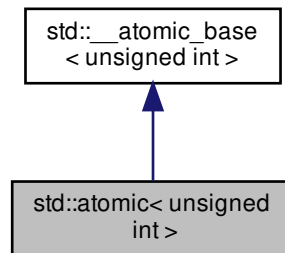
Definition at line 592 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.608 std::atomic< unsigned int > Struct Template Reference

Inheritance diagram for std::atomic< unsigned int >:



#### Public Types

- typedef [\\_\\_atomic\\_base](#)< unsigned int > **\_\_base\_type**
- typedef unsigned int **\_\_integral\_type**

#### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept

- `__int_type operator&= (__int_type __i) noexcept`
- `__int_type operator&= (__int_type __i) volatile noexcept`
- `__int_type operator++ (int) noexcept`
- `__int_type operator++ (int) volatile noexcept`
- `__int_type operator++ () noexcept`
- `__int_type operator++ () volatile noexcept`
- `__int_type operator+= (__int_type __i) noexcept`
- `__int_type operator+= (__int_type __i) volatile noexcept`
- `__int_type operator-- (int) noexcept`
- `__int_type operator-- (int) volatile noexcept`
- `__int_type operator-- () noexcept`
- `__int_type operator-- () volatile noexcept`
- `__int_type operator-= (__int_type __i) noexcept`
- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`

#### 5.608.1 Detailed Description

`template<>`

`struct std::atomic< unsigned int >`

Explicit specialization for unsigned int.

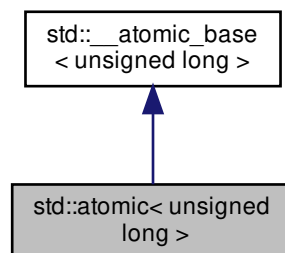
Definition at line 684 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 5.609 std::atomic< unsigned long > Struct Template Reference

Inheritance diagram for `std::atomic< unsigned long >`:



## Public Types

- typedef [\\_\\_atomic\\_base](#)< unsigned long > **\_\_base\_type**
- typedef unsigned long **\_\_integral\_type**

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i)
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

## 5.609.1 Detailed Description

```
template<>
struct std::atomic< unsigned long >
```

Explicit specialization for unsigned long.

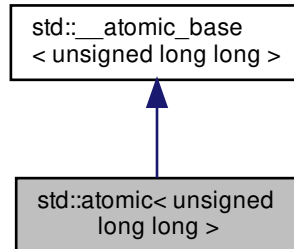
Definition at line 730 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.610 std::atomic&lt; unsigned long long &gt; Struct Template Reference

Inheritance diagram for std::atomic< unsigned long long >:



## Public Types

- typedef `__atomic_base< unsigned long long >` `__base_type`
- typedef `unsigned long long` `__integral_type`

## Public Member Functions

- `atomic` (const `atomic` &)=delete
- constexpr `atomic` (`__integral_type` \_\_i) noexcept
- `__attribute__((always_inline))` void store(`__int_type` \_\_i
- bool `is_lock_free` () const noexcept
- bool `is_lock_free` () const volatile noexcept
- `operator __int_type` () const noexcept
- `operator __int_type` () const volatile noexcept
- `__int_type operator&=` (`__int_type` \_\_i) noexcept
- `__int_type operator&=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator++` (int) noexcept
- `__int_type operator++` (int) volatile noexcept
- `__int_type operator++` () noexcept
- `__int_type operator++` () volatile noexcept
- `__int_type operator+=` (`__int_type` \_\_i) noexcept
- `__int_type operator+=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator--` (int) noexcept
- `__int_type operator--` (int) volatile noexcept
- `__int_type operator--` () noexcept
- `__int_type operator--` () volatile noexcept
- `__int_type operator-=` (`__int_type` \_\_i) noexcept
- `__int_type operator-=` (`__int_type` \_\_i) volatile noexcept
- `atomic` & `operator=` (const `atomic` &)=delete
- `atomic` & `operator=` (const `atomic` &) volatile=delete
- `__int_type operator^=` (`__int_type` \_\_i) noexcept
- `__int_type operator^=` (`__int_type` \_\_i) volatile noexcept
- `__int_type operator|=` (`__int_type` \_\_i) noexcept
- `__int_type operator|=` (`__int_type` \_\_i) volatile noexcept

### 5.610.1 Detailed Description

```
template<>
struct std::atomic< unsigned long long >
```

Explicit specialization for unsigned long long.

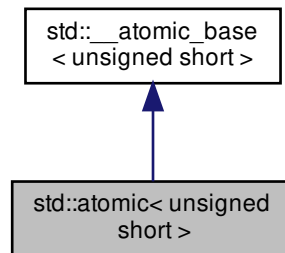
Definition at line 776 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.611 std::atomic< unsigned short > Struct Template Reference

Inheritance diagram for std::atomic< unsigned short >:



#### Public Types

- typedef [\\_\\_atomic\\_base](#)< unsigned short > **\_\_base\_type**
- typedef unsigned short **\_\_integral\_type**

#### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept

- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

#### 5.611.1 Detailed Description

template<>

struct std::atomic< unsigned short >

Explicit specialization for unsigned short.

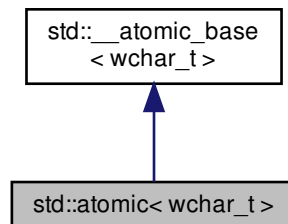
Definition at line 638 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.612 std::atomic< wchar\_t > Struct Template Reference

Inheritance diagram for std::atomic< wchar\_t >:





## Public Types

- typedef [\\_\\_atomic\\_base](#)< wchar\_t > [\\_\\_base\\_type](#)
- typedef wchar\_t [\\_\\_integral\\_type](#)

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **\_\_attribute\_\_** ((\_\_always\_inline\_\_)) void store(\_\_int\_type \_\_i)
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator^=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator|=** (\_\_int\_type \_\_i) volatile noexcept

## 5.612.1 Detailed Description

```
template<>
struct std::atomic< wchar_t >
```

Explicit specialization for wchar\_t.

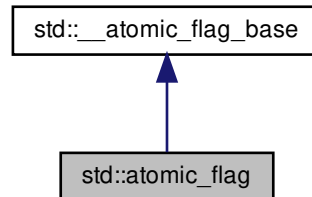
Definition at line 799 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.613 std::atomic\_flag Struct Reference

Inheritance diagram for std::atomic\_flag:



### Public Member Functions

- **atomic\_flag** (const [atomic\\_flag](#) &)=delete
- constexpr **atomic\_flag** (bool \_\_i) noexcept
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &)=delete
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &) volatile=delete

### Public Attributes

- \_\_atomic\_flag\_data\_type **M\_i**

#### 5.613.1 Detailed Description

atomic\_flag

Definition at line 160 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 5.614 std::auto\_ptr<\_Tp> Class Template Reference

### Public Types

- typedef \_Tp [element\\_type](#)

## Public Member Functions

- `auto_ptr` (`element_type` \* \_\_p=0) throw ()
- `auto_ptr` (`auto_ptr` & \_\_a) throw ()
- `template<typename _Tp1 >`  
`auto_ptr` (`auto_ptr`< \_Tp1 > & \_\_a) throw ()
- `auto_ptr` (`auto_ptr_ref`< `element_type` > \_\_ref) throw ()
- `~auto_ptr` ()
- `element_type` \* `get` () const throw ()
- `template<typename _Tp1 >`  
`operator auto_ptr`< \_Tp1 > () throw ()
- `template<typename _Tp1 >`  
`operator auto_ptr_ref`< \_Tp1 > () throw ()
- `element_type` & `operator*` () const throw ()
- `element_type` \* `operator->` () const throw ()
- `auto_ptr` & `operator=` (`auto_ptr` & \_\_a) throw ()
- `template<typename _Tp1 >`  
`auto_ptr` & `operator=` (`auto_ptr`< \_Tp1 > & \_\_a) throw ()
- `auto_ptr` & `operator=` (`auto_ptr_ref`< `element_type` > \_\_ref) throw ()
- `element_type` \* `release` () throw ()
- `void reset` (`element_type` \* \_\_p=0) throw ()

### 5.614.1 Detailed Description

```
template<typename _Tp>
class std::auto_ptr<_Tp>
```

A simple smart pointer providing strict ownership semantics.

The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS 127. `auto_ptr`<> conversion issues These resolutions have all been incorporated.

Definition at line 89 of file `auto_ptr.h`.

## 5.614.2 Member Typedef Documentation

## 5.614.2.1 element\_type

```
template<typename _Tp>
typedef _Tp std::auto_ptr< _Tp >::element_type
```

The pointed-to type.

Definition at line 96 of file auto\_ptr.h.

## 5.614.3 Constructor &amp; Destructor Documentation

## 5.614.3.1 auto\_ptr() [1/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
    element_type * __p = 0 ) throw ( )    [inline], [explicit]
```

An auto\_ptr is usually constructed from a raw pointer.

## Parameters

$\leftrightarrow$ __p	A pointer (defaults to NULL).
--------------------------	-------------------------------

This object now *owns* the object pointed to by \_\_p.

Definition at line 105 of file auto\_ptr.h.

## 5.614.3.2 auto\_ptr() [2/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
    auto_ptr< _Tp > & __a ) throw ( )    [inline]
```

An auto\_ptr can be constructed from another auto\_ptr.

## Parameters

$\leftrightarrow$ __a	Another auto_ptr of the same type.
--------------------------	------------------------------------

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 114 of file `auto_ptr.h`.

#### 5.614.3.3 `auto_ptr()` [3/4]

```
template<typename _Tp>
template<typename _Tp1 >
std::auto_ptr< _Tp >::auto_ptr (
    auto_ptr< _Tp1 > & __a ) throw ( )    [inline]
```

An `auto_ptr` can be constructed from another `auto_ptr`.

##### Parameters

<code>__a</code>	Another <code>auto_ptr</code> of a different but related type.
------------------	--

A pointer-to-`Tp1` must be convertible to a pointer-to-`Tp/element_type`.

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 127 of file `auto_ptr.h`.

#### 5.614.3.4 `~auto_ptr()`

```
template<typename _Tp>
std::auto_ptr< _Tp >::~~auto_ptr ( )    [inline]
```

When the `auto_ptr` goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get()` is `NULL`), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 172 of file `auto_ptr.h`.

#### 5.614.3.5 `auto_ptr()` [4/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
    auto_ptr_ref< element_type > __ref ) throw ( )    [inline]
```

Automatic conversions.

These operations are supposed to convert an `auto_ptr` into and from an `auto_ptr_ref` automatically as needed. This would allow constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(.....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(.....);
```

But it doesn't work, and won't be fixed. For further details see <http://cplusplus.github.io/LWG/G/lwg-closed.html#463>

Definition at line 266 of file `auto_ptr.h`.

#### 5.614.4 Member Function Documentation

##### 5.614.4.1 get()

```
template<typename _Tp>
element_type* std::auto_ptr< _Tp >::get (
    void ) const throw ( )    [inline]
```

Bypassing the smart pointer.

##### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

##### Note

This auto\_ptr still owns the memory.

Definition at line 213 of file auto\_ptr.h.

##### 5.614.4.2 operator\*()

```
template<typename _Tp>
element_type& std::auto_ptr< _Tp >::operator* ( ) const throw ( )    [inline]
```

Smart pointer dereferencing.

If this auto\_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 183 of file auto\_ptr.h.

##### 5.614.4.3 operator->()

```
template<typename _Tp>
element_type* std::auto_ptr< _Tp >::operator-> ( ) const throw ( )    [inline]
```

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.


Definition at line 196 of file auto\_ptr.h.

##### 5.614.4.4 operator=() [1/2]

```
template<typename _Tp>
auto_ptr& std::auto_ptr< _Tp >::operator= (
    auto_ptr< _Tp > & __a ) throw ( )    [inline]
```

auto\_ptr assignment operator.

**Parameters**

	Another auto_ptr of the same type.
<b><i>_a</i></b>	

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 138 of file `auto_ptr.h`.


References `std::auto_ptr<_Tp>::reset()`.

**5.614.4.5 operator=()** [2/2]

```
template<typename _Tp>
template<typename _Tp1 >
auto_ptr& std::auto_ptr<_Tp>::operator= (
    auto_ptr<_Tp1> & __a ) throw ( )    [inline]
```

auto\_ptr assignment operator.

**Parameters**

	Another auto_ptr of a different but related type.
<b><i>_a</i></b>	

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 156 of file `auto_ptr.h`.

References `std::auto_ptr<_Tp>::reset()`.

**5.614.4.6 release()**

```
template<typename _Tp>
element_type* std::auto_ptr<_Tp>::release ( ) throw ( )    [inline]
```

Bypassing the smart pointer.

**Returns**

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

**Note**

This auto\_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 227 of file auto\_ptr.h.

**5.614.4.7 reset()**

```
template<typename _Tp>
void std::auto_ptr< _Tp >::reset (
    element_type * __p = 0 ) throw ( )    [inline]
```

Forcibly deletes the managed object.

**Parameters**

<code>__p</code>	A pointer (defaults to NULL).
------------------	-------------------------------

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.

Definition at line 242 of file auto\_ptr.h.

Referenced by std::auto\_ptr< \_Tp >::operator=().

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

**5.615 std::auto\_ptr\_ref< \_Tp1 > Struct Template Reference****Public Member Functions**

- **auto\_ptr\_ref** (\_Tp1 \* \_\_p)

**Public Attributes**

- `_Tp1 * _M_ptr`



### 5.615.1 Detailed Description

```
template<typename _Tp1>
struct std::auto_ptr_ref< _Tp1 >
```

A wrapper class to provide `auto_ptr` with reference semantics. For example, an `auto_ptr` can be assigned (or constructed from) the result of a function which returns an `auto_ptr` by value.

All the `auto_ptr_ref` stuff should happen behind the scenes.

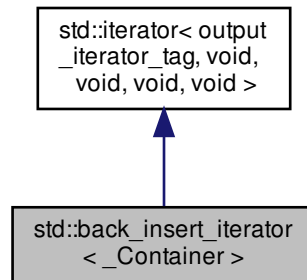
Definition at line 48 of file `auto_ptr.h`.

The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

### 5.616 `std::back_insert_iterator< _Container >` Class Template Reference

Inheritance diagram for `std::back_insert_iterator< _Container >`:



#### Public Types

- typedef `_Container` [container\\_type](#)
- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- [back\\_insert\\_iterator](#) ([\\_Container](#) &\_\_x)
- [back\\_insert\\_iterator](#) & [operator\\*](#) ()
- [back\\_insert\\_iterator](#) & [operator++](#) ()
- [back\\_insert\\_iterator](#) [operator++](#) (int)
- [back\\_insert\\_iterator](#) & [operator=](#) (const typename [\\_Container](#)::value\_type &\_\_value)
- [back\\_insert\\_iterator](#) & [operator=](#) (typename [\\_Container](#)::value\_type &&\_\_value)

### Protected Attributes

- [\\_Container](#) \* **container**

#### 5.616.1 Detailed Description

```
template<typename _Container>
class std::back_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

Definition at line 455 of file `bits/stl_iterator.h`.

#### 5.616.2 Member Typedef Documentation

##### 5.616.2.1 container\_type

```
template<typename _Container >
typedef _Container std::back\_insert\_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

Definition at line 463 of file `bits/stl_iterator.h`.

##### 5.616.2.2 difference\_type

```
typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

### 5.616.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >↵  
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

### 5.616.2.4 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

### 5.616.2.5 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

### 5.616.2.6 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

## 5.616.3 Constructor & Destructor Documentation

### 5.616.3.1 back\_insert\_iterator()

```
template<typename _Container >  
std::back_insert_iterator< _Container >::back_insert_iterator (   
    _Container & __x ) [inline], [explicit]
```

The only way to create this iterator is with a container.

Definition at line 467 of file bits/stl\_iterator.h.

#### 5.616.4 Member Function Documentation

##### 5.616.4.1 operator\*()

```
template<typename _Container >
back_insert_iterator& std::back_insert_iterator< _Container >::operator* ( ) [inline]
```

Simply returns \*this.

Definition at line 506 of file bits/stl\_iterator.h.

##### 5.616.4.2 operator++() [1/2]

```
template<typename _Container >
back_insert_iterator& std::back_insert_iterator< _Container >::operator++ ( ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 511 of file bits/stl\_iterator.h.

##### 5.616.4.3 operator++() [2/2]

```
template<typename _Container >
back_insert_iterator std::back_insert_iterator< _Container >::operator++ (
    int ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 516 of file bits/stl\_iterator.h.

##### 5.616.4.4 operator=()

```
template<typename _Container >
back_insert_iterator& std::back_insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value ) [inline]
```

###### Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> .
----------------------	---

### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

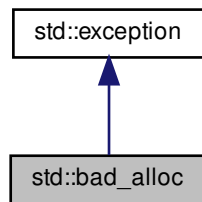
Definition at line 490 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 5.617 std::bad\_alloc Class Reference

Inheritance diagram for `std::bad_alloc`:



### Public Member Functions

- virtual const char \* [what](#) () const throw ()

#### 5.617.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 54 of file `new`.

#### 5.617.2 Member Function Documentation

5.617.2.1 `what()`

```
virtual const char* std::bad_alloc::what ( ) const throw ( ) [virtual]
```

Returns a C-style character string describing the general cause of the current error.

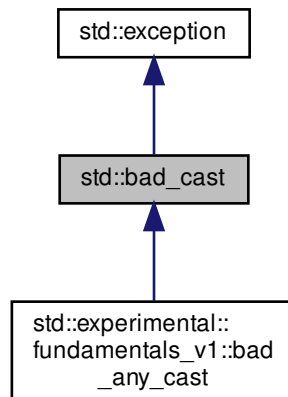
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [new](#)

5.618 `std::bad_cast` Class Reference

Inheritance diagram for `std::bad_cast`:



## Public Member Functions

- virtual const char \* [what](#) ( ) const noexcept

## 5.618.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 187 of file `typeinfo`.

## 5.618.2 Member Function Documentation

### 5.618.2.1 what()

```
virtual const char* std::bad_cast::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

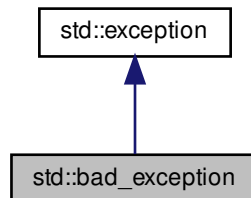
Reimplemented in [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.619 std::bad\_exception Class Reference

Inheritance diagram for std::bad\_exception:



### Public Member Functions

- virtual const char \* [what](#) ( ) const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

### 5.619.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 46 of file exception.

## 5.619.2 Member Function Documentation

5.619.2.1 `what()`

```
virtual const char* std::bad_exception::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

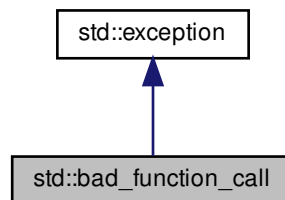
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [exception](#)

5.620 `std::bad_function_call` Class Reference

Inheritance diagram for `std::bad_function_call`:



## Public Member Functions

- `const char * what () const noexcept`

## 5.620.1 Detailed Description

Exception class thrown when class template function's `operator()` is called with an empty target.

Definition at line 56 of file `std_function.h`.



## 5.620.2 Member Function Documentation

### 5.620.2.1 `what()`

```
const char* std::bad_function_call::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

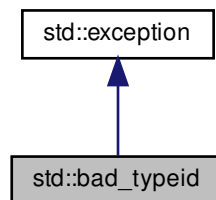
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

## 5.621 `std::bad_typeid` Class Reference

Inheritance diagram for `std::bad_typeid`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

### 5.621.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 204 of file `typeinfo`.

## 5.621.2 Member Function Documentation

## 5.621.2.1 what()

```
virtual const char* std::bad_typeid::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

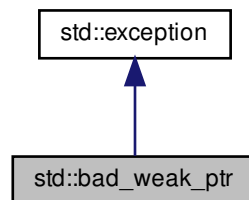
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.622 std::bad\_weak\_ptr Class Reference

Inheritance diagram for std::bad\_weak\_ptr:



## Public Member Functions

- virtual char const \* [what](#) ( ) const noexcept

## 5.622.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

Definition at line 73 of file `shared_ptr_base.h`.

## 5.622.2 Member Function Documentation

### 5.622.2.1 what()

```
virtual char const* std::bad_weak_ptr::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

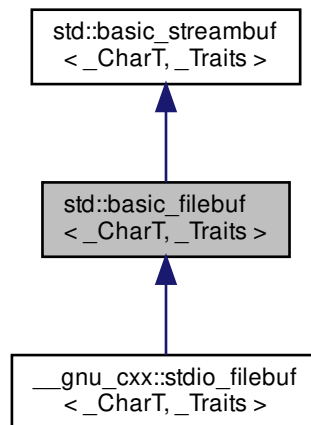
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 5.623 std::basic\_filebuf< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_filebuf< \_CharT, \_Traits >:



### Public Types

- typedef [codecvt](#)< char\_type, char, \_\_state\_type > **\_\_codecvt\_type**
- typedef \_\_basic\_file< char > **\_\_file\_type**
- typedef [basic\\_filebuf](#)< char\_type, traits\_type > **\_\_filebuf\_type**
- typedef traits\_type::state\_type **\_\_state\_type**
- typedef [basic\\_streambuf](#)< char\_type, traits\_type > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

## Public Member Functions

- [basic\\_filebuf](#) ()
  - **basic\_filebuf** (const [basic\\_filebuf](#) &)=delete
  - **basic\_filebuf** ([basic\\_filebuf](#) &&)
  - virtual [~basic\\_filebuf](#) ()
  - [\\_\\_filebuf\\_type](#) \* [close](#) ()
  - [locale](#) [getloc](#) () const
  - [streamsize](#) [in\\_avail](#) ()
  - bool [is\\_open](#) () const throw ()
  - [\\_\\_filebuf\\_type](#) \* [open](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode)
  - [\\_\\_filebuf\\_type](#) \* [open](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode)
  - [basic\\_filebuf](#) & **operator=** (const [basic\\_filebuf](#) &)=delete
  - [basic\\_filebuf](#) & **operator=** ([basic\\_filebuf](#) &&)
  - [locale](#) [pubimbue](#) (const [locale](#) & \_\_loc)
  - int\_type [sbumpc](#) ()
  - int\_type [sgetc](#) ()
  - [streamsize](#) [sgetn](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
  - int\_type [snextc](#) ()
  - int\_type [sputbackc](#) (char\_type \_\_c)
  - int\_type [sputc](#) (char\_type \_\_c)
  - [streamsize](#) [sputn](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
  - int\_type [sungetc](#) ()
  - void **swap** ([basic\\_filebuf](#) &)
- 
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
  - pos\_type [pubseekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - pos\_type [pubseekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - int [pubsync](#) ()

## Protected Member Functions

- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
- void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
- void [\\_M\\_allocate\\_internal\\_buffer](#) ()
- bool [\\_M\\_convert\\_to\\_external](#) (char\_type \*, [streamsize](#))
- void [\\_M\\_create\\_pback](#) ()
- void [\\_M\\_destroy\\_internal\\_buffer](#) () throw ()
- void [\\_M\\_destroy\\_pback](#) () throw ()
- int [\\_M\\_get\\_ext\\_pos](#) (\_\_state\_type & \_\_state)
- pos\_type [\\_M\\_seek](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, \_\_state\_type \_\_state)
- void [\\_M\\_set\\_buffer](#) ([streamsize](#) \_\_off)
- bool [\\_M\\_terminate\\_output](#) ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) & \_\_loc)
- virtual void [imbue](#) (const [locale](#) & \_\_loc \_\_attribute\_\_((\_\_unused\_\_)))
- virtual int\_type **overflow** (int\_type \_\_c= [\\_Traits::eof](#)())

- virtual int\_type [overflow](#) (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
- virtual int\_type **pbackfail** (int\_type \_\_c=\_Traits::eof())
- virtual int\_type [pbackfail](#) (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual pos\_type [seekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual pos\_type [seekpos](#) (pos\_type \_\_pos, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- void [setg](#) (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
- void [setp](#) (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
- virtual [streamsize](#) [showmanyc](#) ()
- void **swap** ([basic\\_streambuf](#) &\_\_sb)
- virtual int [sync](#) ()
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- virtual [streamsize](#) [xsgetn](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- virtual [streamsize](#) [xspn](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)

- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const

- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

#### Protected Attributes

- char\_type \* [\\_M\\_buf](#)
- bool [\\_M\\_buf\\_allocated](#)
- [locale](#) [\\_M\\_buf\\_locale](#)
- size\_t [\\_M\\_buf\\_size](#)
- const [\\_\\_codecvt\\_type](#) \* [\\_M\\_codecvt](#)
- char \* [\\_M\\_ext\\_buf](#)
- [streamsize](#) [\\_M\\_ext\\_buf\\_size](#)
- char \* [\\_M\\_ext\\_end](#)
- const char \* [\\_M\\_ext\\_next](#)
- [\\_\\_file\\_type](#) [\\_M\\_file](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- [\\_\\_c\\_lock](#) [\\_M\\_lock](#)
- [ios\\_base::openmode](#) [\\_M\\_mode](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)

- char\_type \* [\\_M\\_out\\_end](#)
- bool [\\_M\\_reading](#)
- \_\_state\_type [\\_M\\_state\\_beg](#)
- \_\_state\_type [\\_M\\_state\\_cur](#)
- \_\_state\_type [\\_M\\_state\\_last](#)
- bool [\\_M\\_writing](#)

- char\_type [\\_M\\_pback](#)
- char\_type \* [\\_M\\_pback\\_cur\\_save](#)
- char\_type \* [\\_M\\_pback\\_end\\_save](#)
- bool [\\_M\\_pback\\_init](#)

#### Friends

- class [ios\\_base](#)

#### 5.623.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_filebuf< _CharT, _Traits >
```

The actual work of input and output (for files).

#### Template Parameters

<a href="#">_CharT</a>	Type of character stream.
<a href="#">_Traits</a>	Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> .

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Requirements on traits\_type, specific to this class:

- traits\_type::pos\_type must be `fpos<traits_type::state_type>`
- traits\_type::off\_type must be `streamoff`
- traits\_type::state\_type must be Assignable and DefaultConstructible,
- traits\_type::state\_type() must be the initial state for `codecvt`.

Definition at line 80 of file `fstream`.

## 5.623.2 Constructor & Destructor Documentation

### 5.623.2.1 `basic_filebuf()`

```
template<typename _CharT, typename _Traits >
std::basic_filebuf< _CharT, _Traits >::basic_filebuf ( )
```

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 80 of file `fstream.tcc`.

### 5.623.2.2 `~basic_filebuf()`

```
template<typename _CharT, typename _Traits>
virtual std::basic_filebuf< _CharT, _Traits >::~~basic_filebuf ( ) [inline], [virtual]
```

The destructor closes the file first.

Definition at line 246 of file `fstream`.

## 5.623.3 Member Function Documentation

### 5.623.3.1 `_M_create_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback ( ) [inline], [protected]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 199 of file `fstream`.

### 5.623.3.2 `_M_destroy_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback ( ) throw ( ) [inline], [protected]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 216 of file `fstream`.

## 5.623.3.3 \_M\_set\_buffer()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
    streamsize __off ) [inline], [protected]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

\_\_off == egptr() - eback() upon underflow/uflow (**read** mode); \_\_off == 0 upon overflow (**write** mode); \_\_off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: epptr() - pbase() == \_M\_buf\_size - 1, since \_M\_buf\_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 443 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::close().

## 5.623.3.4 close()

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
( )
```

Closes the currently associated file.

## Returns

this on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, overflow(eof) is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 213 of file fstream.tcc.

## 5.623.3.5 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 489 of file streambuf.



#### 5.623.3.6 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

#### 5.623.3.7 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 5.623.3.8 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the read position.

##### Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

#### 5.623.3.9 getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

#### 5.623.3.10 gptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

#### 5.623.3.11 imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale &__loc __attribute__((unused)) ) [inline], [protected], [virtual],
[inherited]
```

Changes translations.

**Parameters**

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Definition at line 583 of file streambuf.

**5.623.3.12 in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file streambuf.

**5.623.3.13 is\_open()**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open ( ) const throw ( ) [inline]
```

Returns true if the external file is open.

Definition at line 260 of file fstream.

**5.623.3.14 open() [1/2]**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode )
```

Opens an external file.

## Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

## Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

ios_base Flag combination					stdio equivalent
binary	in	out	trunc	app	
-----					
		+			w
		+		+	a
				+	a
		+	+		w
	+				r
	+	+			r+
	+	+	+		w+
	+	+		+	a+
	+			+	a+
-----					
+		+			wb
+		+		+	ab
+				+	ab
+		+	+		wb
+	+				rb
+	+	+			r+b
+	+	+	+		w+b
+	+	+		+	a+b
+	+			+	a+b
-----					

Definition at line 179 of file fstream.tcc.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

## 5.623.3.15 open() [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type* std::basic_filebuf< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode ) [inline]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

**Returns**

`this` on success, `NULL` on failure

Definition at line 315 of file `fstream`.

**5.623.3.16 overflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 775 of file `streambuf`.

## 5.623.3.17 pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
    int_type __c __attribute__((__unused__)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

<code>_c</code>	The character to be inserted back into the sequence.
-----------------	--

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 731 of file `streambuf`.

**5.623.3.18 pbase()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pbase ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

**5.623.3.19 pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the write position.

## Parameters

<code>_↵</code>	The delta by which to move.
<code>_n</code>	

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

## 5.623.3.20 pptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

## 5.623.3.21 pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc ) [inline], [inherited]
```

Entry point for imbue().

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.



#### 5.623.3.22 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

##### Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

#### 5.623.3.23 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

##### Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

#### 5.623.3.24 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 5.623.3.25 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]
```

Calls virtual sync function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 5.623.3.26 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 5.623.3.27 `seekoff()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out ) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 798 of file `fstream.tcc`.

**5.623.3.28 seekpos()**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out ) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 858 of file `fstream.tcc`.

**5.623.3.29 setbuf()**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↵
::setbuf (
    char_type * __s,
    streamsize __n ) [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

<code>↵ __s</code>	Pointer to a buffer area.
<code>↵ __n</code>	Size of <code>__s</code> .

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↵html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 769 of file `fstream.tcc`.

## 5.623.3.30 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected], [inherited]
```

Setting the three read area pointers.

## Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

## Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

## 5.623.3.31 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend ) [inline], [protected], [inherited]
```

Setting the three write area pointers.

## Parameters

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

## Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

**5.623.3.32 sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

**5.623.3.33 sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

**5.623.3.34 showmanyc()**

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ( ) [protected], [virtual]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 263 of file `fstream.tcc`.

**5.623.3.35 snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**5.623.3.36 sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c ) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

<code>__c</code>	The character to push back.
------------------	-----------------------------

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**5.623.3.37 sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
    char_type __c ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

<code>__c</code>	A character to output.
------------------	------------------------

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**5.623.3.38 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for all single-character output functions.

## Parameters

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

5.623.3.39 `sungetc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline], [inherited]
```

Moving backwards in the input stream.

## Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

5.623.3.40 `sync()`

```
template<typename _CharT , typename _Traits >
int std::basic_filebuf< _CharT, _Traits >::sync ( ) [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

## Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

## Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 978 of file `fstream.tcc`.



#### 5.623.3.41 uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

#### 5.623.3.42 underflow()

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ( )
[protected], [virtual]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 289 of file `fstream.tcc`.

#### 5.623.3.43 xsgetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
    char_type * __s,
    streamsize __n ) [protected], [virtual]
```

Multiple character extraction.

## Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 635 of file `fstream.tcc`.

## 5.623.3.44 xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
    const char_type * __s,
    streamsize __n ) [protected], [virtual]
```

Multiple character insertion.

## Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 721 of file `fstream.tcc`.

#### 5.623.4 Member Data Documentation

##### 5.623.4.1 `_M_buf`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf [protected]
```

Pointer to the beginning of internal buffer.

Definition at line 136 of file `fstream`.

##### 5.623.4.2 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file `streambuf`.

##### 5.623.4.3 `_M_buf_size`

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size [protected]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 143 of file `fstream`.

##### 5.623.4.4 `_M_ext_buf`

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 178 of file `fstream`.

#### 5.623.4.5 \_M\_ext\_buf\_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected]
```

Size of buffer held by \_M\_ext\_buf.

Definition at line 183 of file fstream.

#### 5.623.4.6 \_M\_ext\_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected]
```

Pointers into the buffer held by \_M\_ext\_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, \_M\_ext\_next corresponds to egptr().

Definition at line 190 of file fstream.

#### 5.623.4.7 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 5.623.4.8 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

#### 5.623.4.9 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 5.623.4.10 `_M_mode`

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current filebuf.

Definition at line 121 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

#### 5.623.4.11 `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 5.623.4.12 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 5.623.4.13 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file streambuf.

#### 5.623.4.14 \_M\_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 164 of file fstream.

#### 5.623.4.15 \_M\_pback\_cur\_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 165 of file fstream.

#### 5.623.4.16 \_M\_pback\_end\_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 166 of file fstream.

#### 5.623.4.17 `_M_pback_init`

```
template<typename _CharT, typename _Traits>
bool std::basic\_filebuf< _CharT, _Traits >::_M_pback_init [protected]
```

Necessary bits for putback buffer management.

#### Note

pbacks of over one character are not currently supported.

Definition at line 167 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

#### 5.623.4.18 `_M_reading`

```
template<typename _CharT, typename _Traits>
bool std::basic\_filebuf< _CharT, _Traits >::_M_reading [protected]
```

`_M_reading == false` && `_M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true` && `_M_writing == true` is unused.

Definition at line 155 of file `fstream`.

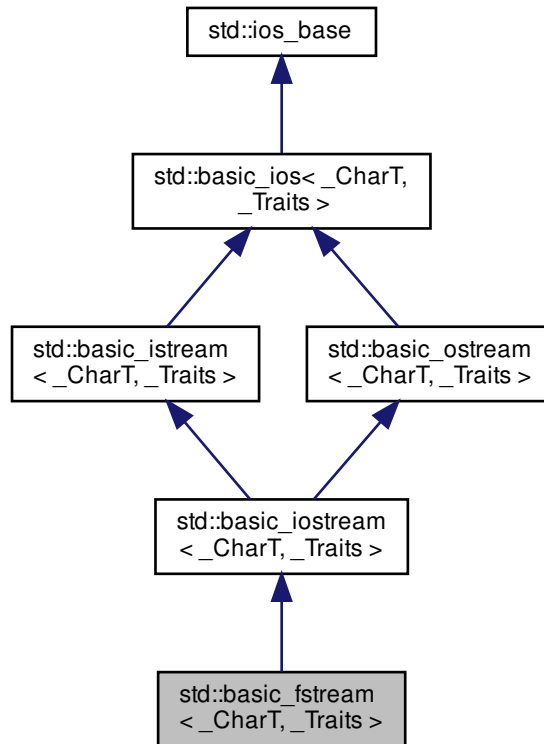
Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

The documentation for this class was generated from the following files:

- [fstream](#)
- [fstream.tcc](#)

## 5.624 std::basic\_fstream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_fstream< \_CharT, \_Traits >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< char_type, traits_type > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`



- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)
  - typedef traits\_type::int\_type **int\_type**
  - typedef int **io\_state**
  - typedef [\\_ios\\_iostate](#) [iostate](#)
  - typedef traits\_type::off\_type **off\_type**
  - typedef int **open\_mode**
  - typedef [\\_ios\\_Openmode](#) [openmode](#)
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef int **seek\_dir**
  - typedef [\\_ios\\_Seekdir](#) [seekdir](#)
  - typedef [std::streamoff](#) **streamoff**
  - typedef [std::streampos](#) **streampos**
  - typedef [\\_Traits](#) **traits\_type**
- 
- typedef [num\\_put](#)< [\\_CharT](#), [ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_fstream](#) ()
- [basic\\_fstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- [basic\\_fstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in|ios\\_base::out](#))
- **basic\_fstream** (const [basic\\_fstream](#) &)=delete
- **basic\_fstream** ([basic\\_fstream](#) &&\_\_rhs)
- [~basic\\_fstream](#) ()
- template<typename [\\_ValueT](#) >  
[basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & **\_M\_extract** ([\\_ValueT](#) &\_\_v)
- const [locale](#) & **\_M\_getloc** () const
- template<typename [\\_ValueT](#) >  
[basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & **\_M\_insert** ([\\_ValueT](#) \_\_v)
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool **bad** () const
- void **clear** ([iostate](#) \_\_state=[goodbit](#))
- void **close** ()
- [basic\\_ios](#) & **copyfmt** (const [basic\\_ios](#) &\_\_rhs)
- bool **eof** () const
- [iostate](#) **exceptions** () const
- void **exceptions** ([iostate](#) \_\_except)
- bool **fail** () const
- char\_type **fill** () const
- char\_type **fill** (char\_type \_\_ch)
- [fmtflags](#) **flags** () const
- [fmtflags](#) **flags** ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & **flush** ()
- [streamsize](#) **gcount** () const
- template<>  
[basic\\_istream](#)< char > & **getline** (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)

- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `locale getloc () const`
  - `bool good () const`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `bool is_open ()`
  - `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __default) const`
  - `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (__streambuf_type *__sb)`
  - `basic_fstream & operator= (const basic_fstream &)=delete`
  - `basic_fstream & operator= (basic_fstream &&__rhs)`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `__filebuf_type * rdbuf () const`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void swap (basic_fstream &__rhs)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`

- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
  - `__istream_type & operator>> (short &__n)`
  - `__istream_type & operator>> (unsigned short &__n)`
  - `__istream_type & operator>> (int &__n)`
  - `__istream_type & operator>> (unsigned int &__n)`
  - `__istream_type & operator>> (long &__n)`
  - `__istream_type & operator>> (unsigned long &__n)`
  - `__istream_type & operator>> (long long &__n)`
  - `__istream_type & operator>> (unsigned long long &__n)`
- 
- `__istream_type & operator>> (float &__f)`
  - `__istream_type & operator>> (double &__f)`
  - `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`

- [\\_\\_istream\\_type](#) & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - bool [operator!](#) () const
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { **[\\_S\\_local\\_word\\_size](#)** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
[\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
[\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_iostream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 5.624.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_fstream< _CharT, _Traits >
```

Controlling input and output for files.

### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This class supports reading from and writing to named files, using the inherited functions from `std::basic_iostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 927 of file `fstream`.

## 5.624.2 Member Typedef Documentation

### 5.624.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

### 5.624.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 506 of file `ios_base.h`.

### 5.624.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 323 of file `ios_base.h`.

### 5.624.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 398 of file `ios_base.h`.



#### 5.624.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

#### 5.624.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

### 5.624.3 Member Enumeration Documentation

#### 5.624.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.624.4 Constructor &amp; Destructor Documentation

## 5.624.4.1 basic\_fstream() [1/3]

```
template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream ( ) [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 954 of file `fstream`.

## 5.624.4.2 basic\_fstream() [2/3]

```
template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream (
    const char * __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [explicit]
```

Create an input/output file stream.

## Parameters

<code>__s</code>	Null terminated string specifying the filename.
<code>__mode</code>	Open file in specified mode (see <code>std::ios_base</code> ).

Definition at line 964 of file `fstream`.

## 5.624.4.3 basic\_fstream() [3/3]

```
template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [explicit]
```

Create an input/output file stream.

## Parameters

<code>__s</code>	Null terminated string specifying the filename.
<code>__mode</code>	Open file in specified mode (see <code>std::ios_base</code> ).

Definition at line 979 of file fstream.

#### 5.624.4.4 ~basic\_fstream()

```
template<typename _CharT, typename _Traits >
std::basic_fstream< _CharT, _Traits >::~~basic_fstream ( ) [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 1014 of file fstream.

### 5.624.5 Member Function Documentation

#### 5.624.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

#### 5.624.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

<code>_↵ _s</code>	The array to insert.
<code>_↵ _n</code>	Maximum number of characters to insert.

Definition at line 311 of file ostream.

**5.624.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**5.624.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

#### 5.624.5.5 close()

```
template<typename _CharT, typename _Traits >
void std::basic_fstream< _CharT, _Traits >::close ( ) [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 1129 of file `fstream`.

#### 5.624.5.6 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of `__rhs` into this.

##### Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

##### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

#### 5.624.5.7 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

## 5.624.5.8 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.624.5.9 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file basic\_ios.h.

#### 5.624.5.10 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator bool(), std::basic\_ios< char, \_Traits >::operator!(), and std::basic\_ios< \_CharType >::value().

#### 5.624.5.11 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::basic\_ios< char, \_Traits >::fill(), and std::operator<<().

#### 5.624.5.12 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

##### Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

**5.624.5.13 flags()** [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**5.624.5.14 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file ios\_base.h.



**5.624.5.15 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

**5.624.5.16 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file `istream`.

**5.624.5.17 get()** [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets `failbit` and returns `traits::eof()`.

Definition at line 244 of file `istream.tcc`.

**5.624.5.18 get()** [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c ) [inherited]
```

Simple extraction.

## Parameters

<code>__c</code>	The character in which to store data.
------------------	---------------------------------------

## Returns

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file `istream.tcc`.

## 5.624.5.19 get() [3/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Simple multiple-character extraction.

## Parameters

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>__s</code> .
<code>__delim</code>	A "stop" character.

## Returns

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

**5.624.5.20 get()** [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>s</code> .

**Returns**

`*this`

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

**5.624.5.21 get()** [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim ) [inherited]
```

Extraction into another streambuf.

**Parameters**

<code>__sb</code>	A streambuf in which to store data.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

**5.624.5.22 get()** [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb ) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

<code>__sb</code>	A streambuf in which to store data.
-------------------	-------------------------------------

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

**5.624.5.23 getline()** [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.
<code>__delim</code>	A "stop" character.

**Returns**

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**5.624.5.24 `getline()`** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

#### 5.624.5.25 `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

#### 5.624.5.26 `getloc()`

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, and `std::ws()`.

#### 5.624.5.27 `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 5.624.5.28 `ignore()` [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim ) [inherited]
```

Discarding characters.

**Parameters**

<code>__n</code>	Number of characters to discard.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**5.624.5.29 ignore()** [2/3]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.

## 5.624.5.30 ignore() [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

## 5.624.5.31 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

## 5.624.5.32 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```



All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

#### 5.624.5.33 `is_open()`

```
template<typename _CharT , typename _Traits >
bool std::basic_fstream< _CharT, _Traits >::is_open ( ) [inline]
```

Wrapper to test for an open file.

##### Returns

`rdbuf() -> is_open()`

Definition at line 1055 of file `fstream`.

#### 5.624.5.34 `yword()`

```
long& std::ios_base::yword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

##### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

##### Returns

A reference to an integer associated with the index.

The `yword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

## 5.624.5.35 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Squeezes characters.

## Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file `basic_ios.h`.

## 5.624.5.36 open() [1/2]

```
template<typename _CharT , typename _Traits >
void std::basic_fstream< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline]
```

Opens an external file.

## Parameters

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 1073 of file `fstream`.

**5.624.5.37** `open()` [2/2]

```
template<typename _CharT, typename _Traits >
void std::basic_fstream< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 1094 of file `fstream`.

**5.624.5.38** `operator bool()`

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.624.5.39** `operator!()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

## 5.624.5.40 operator&lt;&lt;() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(*) (__ostream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

## 5.624.5.41 operator&lt;&lt;() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

## 5.624.5.42 operator&lt;&lt;() [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.


Definition at line 127 of file `ostream`.

## 5.624.5.43 operator&lt;&lt;() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

**5.624.5.44 operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**5.624.5.45 operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    bool __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

## 5.624.5.46 operator&lt;&lt;() [7/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n ) [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

## 5.624.5.47 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

<code>_↵</code> <code>_n</code>	A variable of builtin integral type.
------------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**5.624.5.48 operator<<() [9/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    int __n )    [inherited]
```

Integer arithmetic inserters.

**Parameters**

<code>_↵</code> <code>_n</code>	A variable of builtin integral type.
------------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

**5.624.5.49 operator<<() [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n )    [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

$\leftrightarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

## 5.624.5.50 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

$\leftrightarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 5.624.5.51 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n ) [inline], [inherited]
```

Integer arithmetic inserters.



**Parameters**

$\leftarrow$ $\_n$	A variable of builtin integral type.
-----------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**5.624.5.52 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

$\leftarrow$ $\_$ $\leftarrow$ $\_$ $f$	A variable of builtin floating point type.
---	--

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

**5.624.5.53 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 5.624.5.54 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 5.624.5.55 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p ) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

<code>_↵ _p</code>	A variable of pointer type.
------------------------	-----------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**5.624.5.56 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

**Parameters**

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

**5.624.5.57 operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(*) (__istream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

**5.624.5.58 operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

**5.624.5.59 operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.


Definition at line 131 of file `istream`.

**5.624.5.60 operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 168 of file `istream`.

**5.624.5.61 `operator>>()` [5/17]**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n ) [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

**5.624.5.62 `operator>>()` [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftrightarrow$ _n	A variable of builtin integral type.
-------------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

## 5.624.5.63 operator&gt;&gt;() [7/17]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n ) [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftrightarrow$ _n	A variable of builtin integral type.
-------------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

## 5.624.5.64 operator&gt;&gt;() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>_n</code>	A variable of builtin integral type.
---------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**5.624.5.65 operator>>() [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>_n</code>	A variable of builtin integral type.
---------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**5.624.5.66 operator>>() [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

## 5.624.5.67 operator&gt;&gt;() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

## 5.624.5.68 operator&gt;&gt;() [12/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.



**Parameters**

$\leftarrow$ $\_n$	A variable of builtin integral type.
-----------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**5.624.5.69 `operator>>()` [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

$\leftarrow$ $\_$ $\leftarrow$ $\_$ $f$	A variable of builtin floating point type.
---	--

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**5.624.5.70 `operator>>()` [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

## 5.624.5.71 operator&gt;&gt;() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

## 5.624.5.72 operator&gt;&gt;() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p ) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

<code>_↵ _p</code>	A variable of pointer type.
------------------------	-----------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**5.624.5.73 operator>>() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

**Parameters**

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

## 5.624.5.74 peek()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

## Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

## 5.624.5.75 precision() [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

## 5.624.5.76 precision() [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

## Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of precision().


Definition at line 700 of file ios\_base.h.

**5.624.5.77 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c ) [inherited]
```

Simple insertion.

**Parameters**

	The character to insert.
<code>__c</code>	

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.


Definition at line 149 of file ostream.tcc.

**5.624.5.78 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c ) [inherited]
```

Unextracting a single character.

**Parameters**

	The character to push back into the input stream.
<code>__c</code>	

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**5.624.5.79 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file `ios_base.h`.

**5.624.5.80 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**5.624.5.81 `rdbuf()`** [2/2]

```
template<typename _CharT, typename _Traits >
__filebuf_type* std::basic_fstream< _CharT, _Traits >::rdbuf ( ) const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 1047 of file `fstream`.

**5.624.5.82 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

## 5.624.5.83 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction without delimiters.

## Parameters

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

## Returns

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file istream.tcc.

## 5.624.5.84 readsome()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.



**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

**5.624.5.85 register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.624.5.86 seekg()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos ) [inherited]
```

Changing the current read position.

**Parameters**

<code>__pos</code>	A file position object.
--------------------	-------------------------

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**5.624.5.87 `seekg()`** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current read position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**5.624.5.88 `seekp()`** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
    pos_type __pos ) [inherited]
```

Changing the current write position.

**Parameters**

<code>__pos</code>	A file position object.
--------------------	-------------------------

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

**5.624.5.89 seekp()** [2/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current write position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

**5.624.5.90 setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>().

**5.624.5.91 setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file ios\_base.h.

**5.624.5.92 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::ws()`.

#### 5.624.5.93 `sync()`

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

##### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

##### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

#### 5.624.5.94 `sync_with_stdio()`

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**5.624.5.95 tellg()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, in).

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount(). At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 825 of file istream.tcc.

**5.624.5.96 tellp()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ( )
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, out).

Definition at line 237 of file ostream.tcc.

**5.624.5.97** `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**5.624.5.98** `tie()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (
    basic_ostream<_CharT, _Traits> * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**5.624.5.99** `unget()`

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**5.624.5.100 unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

**5.624.5.101 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

**Parameters**

<code>__c</code>	The character to widen.
------------------	-------------------------



**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

**Returns the result of**

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**5.624.5.102 width() [1/2]**

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::num_put<_CharT, _OutItter>::do_put()`.

**5.624.5.103 width() [2/2]**

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 723 of file ios\_base.h.

#### 5.624.5.104 write()

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n ) [inherited]
```

Character string insertion.

##### Parameters

<code>__s</code>	The array to insert.
<code>__n</code>	Maximum number of characters to insert.

##### Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

#### 5.624.5.105 xalloc()

```
static int std::ios_base::xalloc ( ) throw ( ) [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.624.6 Member Data Documentation

### 5.624.6.1 `_M_gcount`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, and `std::basic_istream< char >::unget()`.

### 5.624.6.2 `adjustfield`

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outlter >::do_put()`.

### 5.624.6.3 `app`

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

### 5.624.6.4 `ate`

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

#### 5.624.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), and std::operator<<().

#### 5.624.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 381 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< char >::operator<<().

#### 5.624.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.624.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.624.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.624.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.624.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file `ios_base.h`.

#### 5.624.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.624.6.13 eofbit

```
const ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::eof()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_istream< char >::unget()`, and `std::ws()`.

#### 5.624.6.14 failbit

```
const ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry()`.

#### 5.624.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file `ios_base.h`.

#### 5.624.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

#### 5.624.6.17 goodbit

```
const iosstate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_Inlter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

#### 5.624.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_Inlter >::do\_get(), std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::basic\_ostream< char >::operator<<().

#### 5.624.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for ifstream and fstream.

Definition at line 443 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

#### 5.624.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 340 of file ios\_base.h.

#### 5.624.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

#### 5.624.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::operator<<().

#### 5.624.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for ofstream and fstream.

Definition at line 446 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos().

#### 5.624.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.



**5.624.6.25 scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

**5.624.6.26 showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.624.6.27 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

**5.624.6.28 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

**5.624.6.29 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

#### 5.624.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 449 of file `ios_base.h`.

#### 5.624.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file `ios_base.h`.

#### 5.624.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file `ios_base.h`.

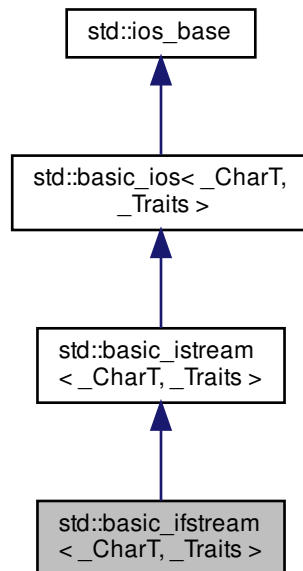
Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

- [fstream](#)

## 5.625 std::basic\_ifstream<\_CharT, \_Traits> Class Template Reference

Inheritance diagram for std::basic\_ifstream<\_CharT, \_Traits>:



### Public Types

- typedef [ctype](#)<\_CharT> **\_\_ctype\_type**
- typedef [basic\\_filebuf](#)<char\_type, traits\_type> **\_\_filebuf\_type**
- typedef [basic\\_ios](#)<\_CharT, \_Traits> **\_\_ios\_type**
- typedef [basic\\_istream](#)<char\_type, traits\_type> **\_\_istream\_type**
- typedef [num\\_get](#)<\_CharT, [istreambuf\\_iterator](#)<\_CharT, \_Traits>> **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)<\_CharT, \_Traits> **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags **fmtflags**
- typedef traits\_type::int\_type **int\_type**
- typedef int **io\_state**
- typedef \_ios\_istate **istate**
- typedef traits\_type::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_ios\_Openmode **openmode**
- typedef traits\_type::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_ios\_Seekdir **seekdir**

- typedef [std::streamoff](#) **streamoff**
  - typedef [std::streampos](#) **streampos**
  - typedef [\\_Traits](#) **traits\_type**
- 
- typedef [num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >](#) **\_\_num\_put\_type**

### Public Member Functions

- [basic\\_ifstream](#) ()
- [basic\\_ifstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- **basic\_ifstream** (const [basic\\_ifstream](#) &)=delete
- **basic\_ifstream** ([basic\\_ifstream](#) && \_\_rhs)
- [~basic\\_ifstream](#) ()
- template<typename \_ValueT >  
[basic\\_istream< \\_CharT, \\_Traits > & \\_M\\_extract](#) (\_ValueT & \_\_v)
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream< char > & getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- template<>  
[basic\\_istream< wchar\\_t > & getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>  
[basic\\_istream< char > & ignore](#) ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream< char > & ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- template<>  
[basic\\_istream< wchar\\_t > & ignore](#) ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream< wchar\\_t > & ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) & \_\_loc)
- bool [is\\_open](#) ()

- `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __default) const`
  - `void open (const char * __s, ios_base::openmode __mode=ios_base::in)`
  - `void open (const std::string & __s, ios_base::openmode __mode=ios_base::in)`
  - `basic_ifstream & operator= (const basic_ifstream &)=delete`
  - `basic_ifstream & operator= (basic_ifstream && __rhs)`
  - `__istream_type & operator>> (void *& __p)`
  - `__istream_type & operator>> (__streambuf_type * __sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `__filebuf_type * rdbuf () const`
  - `iosate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosate __state)`
  - `void swap (basic_ifstream & __rhs)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool & __n)`
- `__istream_type & operator>> (short & __n)`
- `__istream_type & operator>> (unsigned short & __n)`
- `__istream_type & operator>> (int & __n)`
- `__istream_type & operator>> (unsigned int & __n)`
- `__istream_type & operator>> (long & __n)`
- `__istream_type & operator>> (unsigned long & __n)`
- `__istream_type & operator>> (long long & __n)`

- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)

- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ()
  - `int_type` [peek](#) ()
  - [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - `int` [sync](#) ()
  - `pos_type` [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - `bool` [operator!](#) () const

### Static Public Member Functions

- static `bool` [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static `int` [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iosstate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iosstate](#) eofbit
- static const [iosstate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iosstate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { **\_S\_local\_word\_size** }

### Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) & \_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_istream\\_type](#) & **\_M\_extract** (\_ValueT & \_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) & \_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits > \* \_\_sb)
- void **move** ([basic\\_ios](#) & \_\_rhs)
- void **move** ([basic\\_ios](#) && \_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits > \* \_\_sb)
- void **swap** ([basic\\_ios](#) & \_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) & \_\_rhs)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 5.625.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ifstream< _CharT, _Traits >
```

Controlling input for files.

## Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 476 of file `fstream`.

## 5.625.2 Member Typedef Documentation



#### 5.625.2.1 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.




Definition at line 89 of file basic\_ios.h.

#### 5.625.2.2 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

##### Parameters

 <code>__e</code>	One of the members of the event enum.
 <code>__b</code>	Reference to the ios_base object.
 <code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 506 of file ios\_base.h.

#### 5.625.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex

- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 323 of file `ios_base.h`.

#### 5.625.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 398 of file `ios_base.h`.

#### 5.625.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

#### 5.625.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

### 5.625.3 Member Enumeration Documentation

#### 5.625.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.625.4 Constructor &amp; Destructor Documentation

## 5.625.4.1 basic\_ifstream() [1/3]

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream ( ) [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 502 of file `fstream`.

## 5.625.4.2 basic\_ifstream() [2/3]

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
    const char * __s,
    ios_base::openmode __mode = ios_base::in ) [inline], [explicit]
```

Create an input file stream.

## Parameters

<code>__s</code>	Null terminated string specifying the filename.
<code>__mode</code>	Open file in specified mode (see <code>std::ios_base</code> ).

`ios_base::in` is automatically included in `__mode`.

Definition at line 513 of file `fstream`.

## 5.625.4.3 basic\_ifstream() [3/3]

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in ) [inline], [explicit]
```

Create an input file stream.

**Parameters**

<code>__s</code>	std::string specifying the filename.
<code>__mode</code>	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in `__mode`.

Definition at line 529 of file `fstream`.

**5.625.4.4 ~basic\_ifstream()**

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream ( ) [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 566 of file `fstream`.

**5.625.5 Member Function Documentation****5.625.5.1 \_M\_getloc()**

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 776 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

#### 5.625.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

#### 5.625.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

##### Parameters

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

#### 5.625.5.4 close()

```
template<typename _CharT , typename _Traits >
void std::basic_ifstream< _CharT, _Traits >::close ( ) [inline]
```

Close the file.

Calls std::basic\_filebuf::close(). If that function fails, failbit is set in the stream's error state.

Definition at line 678 of file fstream.

#### 5.625.5.5 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of \_\_rhs into this.

##### Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

##### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

#### 5.625.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

#### 5.625.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.625.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 5.625.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, `std::basic_ios< char, _Traits >::operator!()`, and `std::regex_traits< _CharType >::value()`.



**5.625.5.10** `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**5.625.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**5.625.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**5.625.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file ios\_base.h.

**5.625.5.14 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_ifstream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

**5.625.5.15** `get()` [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

**5.625.5.16** `get()` [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c ) [inherited]
```

Simple extraction.

**Parameters**

<code>__c</code>	The character in which to store data.
------------------	---------------------------------------

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file istream.tcc.

**5.625.5.17** `get()` [3/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Simple multiple-character extraction.

## Parameters

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>__s</code> .
<code>__delim</code>	A "stop" character.

## Returns

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

5.625.5.18 `get()` [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_ifstream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Simple multiple-character extraction.

## Parameters

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>s</code> .

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

**5.625.5.19 get()** [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim ) [inherited]
```

Extraction into another streambuf.

**Parameters**

<code>__sb</code>	A streambuf in which to store data.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

**5.625.5.20 get()** [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb ) [inline], [inherited]
```

Extraction into another streambuf.

## Parameters

<code>__sb</code>	A streambuf in which to store data.
-------------------	-------------------------------------

## Returns

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

5.625.5.21 `getline()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

String extraction.

## Parameters

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.
<code>__delim</code>	A "stop" character.

## Returns

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file istream.tcc.

**5.625.5.22** `getline()` [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

**5.625.5.23** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**5.625.5.24** `getloc()`

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, and `std::ws()`.

## 5.625.5.25 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 180 of file basic\_ios.h.

Referenced by std::\_\_detail::operator>>(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

## 5.625.5.26 ignore() [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim ) [inherited]
```

Discarding characters.

## Parameters

<code>__n</code>	Number of characters to discard.
<code>__delim</code>	A "stop" character.

## Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file istream.tcc.



**5.625.5.27 ignore()** [2/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 501 of file istream.tcc.

**5.625.5.28 ignore()** [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**5.625.5.29 imbue()**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**5.625.5.30 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**5.625.5.31 is\_open()**

```
template<typename _CharT , typename _Traits >
bool std::basic_ifstream< _CharT, _Traits >::is_open ( ) [inline]
```

Wrapper to test for an open file.

**Returns**

`rdbuf()->is_open()`

Definition at line 607 of file `fstream`.

**5.625.5.32 iword()**

```
long& std::ios_base::iword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

**Parameters**

<code><a href="#">↵</a> _ix</code>	Index into the array.
--	-----------------------

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

**5.625.5.33 narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>[↵](#)

Definition at line 430 of file `basic_ios.h`.

**5.625.5.34 open()** [1/2]

```
template<typename _CharT , typename _Traits >
void std::basic_ifstream< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode = ios_base::in ) [inline]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Calls `std::basic_filebuf::open(s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 625 of file `fstream`.

**5.625.5.35 open()** [2/2]

```
template<typename _CharT , typename _Traits >
void std::basic_ifstream< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::in ) [inline]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 645 of file `fstream`.

**5.625.5.36 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

**5.625.5.37 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**5.625.5.38 operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(*) (__istream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

**5.625.5.39 operator>>() [2/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

**5.625.5.40 operator>>() [3/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file `istream`.

**5.625.5.41 operator>>() [4/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftrightarrow$ _n	A variable of builtin integral type.
-------------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

## 5.625.5.42 operator&gt;&gt;() [5/17]

```
template<typename _CharT, typename _Traits >
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::operator>> (
    short & __n ) [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftrightarrow$ _n	A variable of builtin integral type.
-------------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

## 5.625.5.43 operator&gt;&gt;() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_ifstream< _CharT, _Traits >::operator>> (
    unsigned short & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftrightarrow$ <code>_n</code>	A variable of builtin integral type.
--------------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**5.625.5.44 `operator>>()` [7/17]**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n ) [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftrightarrow$ <code>_n</code>	A variable of builtin integral type.
--------------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**5.625.5.45 `operator>>()` [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

## 5.625.5.46 operator&gt;&gt;() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

## 5.625.5.47 operator&gt;&gt;() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.



**Parameters**

$\leftarrow$ <code>__n</code>	A variable of builtin integral type.
----------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**5.625.5.48 `operator>>()` [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>__n</code>	A variable of builtin integral type.
----------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**5.625.5.49 `operator>>()` [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
$\_n$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

## 5.625.5.50 operator&gt;&gt;() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin floating point type.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$	
$f$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

## 5.625.5.51 operator&gt;&gt;() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**5.625.5.52 operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**5.625.5.53 operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p ) [inline], [inherited]
```

Basic arithmetic extractors.

## Parameters

<code>_p</code>	A variable of pointer type.
-----------------	-----------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

5.625.5.54 `operator>>()` [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

## Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

**5.625.5.55 peek()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

**5.625.5.56 precision()** [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.625.5.57 precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of precision().


Definition at line 700 of file ios\_base.h.

**5.625.5.58 putback()**

```
template<typename _CharT, typename _Traits >
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::putback (
    char_type __c ) [inherited]
```

Unextracting a single character.

**Parameters**

 <code>__c</code>	The character to push back into the input stream.
--	---

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.


Definition at line 719 of file istream.tcc.

**5.625.5.59 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

 <code>__ix</code>	Index into the array.
---	-----------------------

**Returns**

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file `ios_base.h`.

**5.625.5.60 `rdbuf()`** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**5.625.5.61 `rdbuf()`** [2/2]

```
template<typename _CharT , typename _Traits >
__filebuf_type* std::basic_ifstream< _CharT, _Traits >::rdbuf ( ) const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 599 of file `fstream`.

## 5.625.5.62 rdstate()

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See std::ios\_base::iosstate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 137 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, ↵  
\_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< char >::putback(), std::basic\_istream< ↵  
char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

## 5.625.5.63 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction without delimiters.

## Parameters

↵ __s	A character array.
↵ __n	Maximum number of characters to store.

## Returns

\*this

If the stream state is good ( ) , extracts characters and stores them into \_\_s until one of the following happens:

- \_\_n characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to failbit|eofbit.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file istream.tcc.



**5.625.5.64 readsome()**

```
template<typename _CharT, typename _Traits >
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

**5.625.5.65 register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.625.5.66 seekg() [1/2]

```
template<typename _CharT , typename _Traits >
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::seekg (
    pos_type __pos ) [inherited]
```

Changing the current read position.

##### Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

##### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

##### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file istream.tcc.

#### 5.625.5.67 seekg() [2/2]

```
template<typename _CharT , typename _Traits >
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current read position.

##### Parameters

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

##### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

#### 5.625.5.68 `setf()` [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

#### Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

#### 5.625.5.69 `setf()` [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

#### Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

**5.625.5.70 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

**5.625.5.71 sync()**

```
template<typename _CharT , typename _Traits >
int std::basic_ifstream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

### 5.625.5.72 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true )    [static], [inherited]
```

Interaction with the standard C I/O objects.

#### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

### 5.625.5.73 tellg()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void )    [inherited]
```

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

## 5.625.5.74 tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

## 5.625.5.75 tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

## 5.625.5.76 unget()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**5.625.5.77 unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

**5.625.5.78 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

**Parameters**

<code>__c</code>	The character to widen.
------------------	-------------------------

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**5.625.5.79 width()** [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**5.625.5.80 width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.



Definition at line 723 of file ios\_base.h.

#### 5.625.5.81 xalloc()

```
static int std::ios_base::xalloc ( ) throw ( ) [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.625.6 Member Data Documentation

#### 5.625.6.1 \_M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, and `std::basic_istream< char >::unget()`.

#### 5.625.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`.

#### 5.625.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file ios\_base.h.

#### 5.625.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file ios\_base.h.

#### 5.625.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), and std::operator<<().

#### 5.625.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 381 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< char >::operator<<().

#### 5.625.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.625.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.625.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::num_put<_CharT, _OutIter >::do_put()`.

#### 5.625.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 5.625.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.

#### 5.625.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 5.625.6.13 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_Inlter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 5.625.6.14 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_Inlter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

#### 5.625.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

#### 5.625.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

#### 5.625.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

#### 5.625.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.625.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

#### 5.625.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file `ios_base.h`.

#### 5.625.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.625.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::operator<<()`.

#### 5.625.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 446 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`.

**5.625.6.24 right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

**5.625.6.25 scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

**5.625.6.26 showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put().

**5.625.6.27 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

**5.625.6.28 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

#### 5.625.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

#### 5.625.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 449 of file ios\_base.h.

#### 5.625.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

#### 5.625.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

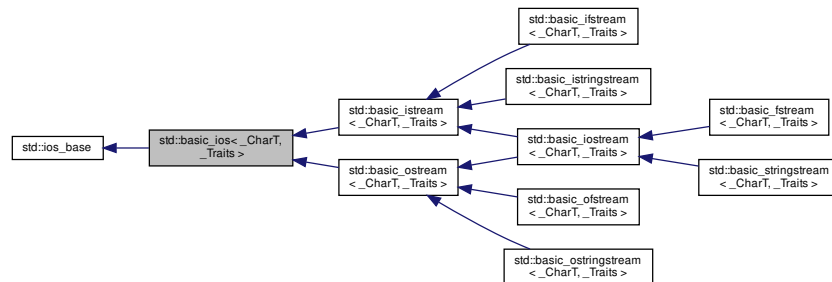
The documentation for this class was generated from the following file:

- [fstream](#)



## 5.626 std::basic\_ios< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_ios< \_CharT, \_Traits >:



### Public Types

- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)
  - typedef int [io\\_state](#)
  - typedef [\\_ios\\_istate](#) [iostate](#)
  - typedef int [open\\_mode](#)
  - typedef [\\_ios\\_Openmode](#) [openmode](#)
  - typedef int [seek\\_dir](#)
  - typedef [\\_ios\\_Seekdir](#) [seekdir](#)
  - typedef [std::streamoff](#) [streamoff](#)
  - typedef [std::streampos](#) [streampos](#)
- 
- typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits::int\\_type](#) [int\\_type](#)
  - typedef [\\_Traits::pos\\_type](#) [pos\\_type](#)
  - typedef [\\_Traits::off\\_type](#) [off\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
- 
- typedef [ctype](#)< [\\_CharT](#) > [\\_\\_ctype\\_type](#)
  - typedef [num\\_put](#)< [\\_CharT](#), [ostreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_put\\_type](#)
  - typedef [num\\_get](#)< [\\_CharT](#), [istreambuf\\_iterator](#)< [\\_CharT](#), [\\_Traits](#) > > [\\_\\_num\\_get\\_type](#)

## Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
  - virtual [~basic\\_ios](#) ()
  - const [locale](#) & [\\_M\\_getloc](#) () const
  - void [\\_M\\_setstate](#) ([iosstate](#) \_\_state)
  - bool [bad](#) () const
  - void [clear](#) ([iosstate](#) \_\_state=[goodbit](#))
  - [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
  - bool [eof](#) () const
  - [iosstate exceptions](#) () const
  - void [exceptions](#) ([iosstate](#) \_\_except)
  - bool [fail](#) () const
  - [char\\_type fill](#) () const
  - [char\\_type fill](#) ([char\\_type](#) \_\_ch)
  - [fmtflags flags](#) () const
  - [fmtflags flags](#) ([fmtflags](#) \_\_fmtfl)
  - [locale getloc](#) () const
  - bool [good](#) () const
  - [locale imbue](#) (const [locale](#) & \_\_loc)
  - long & [iword](#) (int \_\_ix)
  - char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
  - [streamsize precision](#) () const
  - [streamsize precision](#) ([streamsize](#) \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) () const
  - [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
  - [iosstate rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [fmtflags setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iosstate](#) \_\_state)
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type widen](#) (char \_\_c) const
  - [streamsize width](#) () const
  - [streamsize width](#) ([streamsize](#) \_\_wide)
- 
- [operator bool](#) () const
  - bool [operator!](#) () const

## Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { **[\\_S\\_local\\_word\\_size](#)** }

### Protected Member Functions

- [basic\\_ios](#) ()
- **[basic\\_ios](#)** (const [basic\\_ios](#) &)=delete
- void **[\\_M\\_cache\\_locale](#)** (const [locale](#) &\_\_loc)
- void **[\\_M\\_call\\_callbacks](#)** ([event](#) \_\_ev) throw ()
- void **[\\_M\\_dispose\\_callbacks](#)** (void) throw ()
- [\\_Words](#) & **[\\_M\\_grow\\_words](#)** (int \_\_index, bool \_\_iword)
- void **[\\_M\\_init](#)** () throw ()
- void **[\\_M\\_move](#)** ([ios\\_base](#) &) noexcept
- void **[\\_M\\_swap](#)** ([ios\\_base](#) & \_\_rhs) noexcept
- void **[init](#)** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- void **[move](#)** ([basic\\_ios](#) & \_\_rhs)
- void **[move](#)** ([basic\\_ios](#) && \_\_rhs)
- [basic\\_ios](#) & **[operator=](#)** (const [basic\\_ios](#) &)=delete
- void **[set\\_rdbuf](#)** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- void **[swap](#)** ([basic\\_ios](#) & \_\_rhs) noexcept

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 5.626.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ios<_CharT, _Traits>
```

Template class `basic_ios`, virtual base class for all stream classes.

## Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar)`;) are consolidated in this class.

Definition at line 77 of file `iosfwd`.

## 5.626.2 Member Typedef Documentation

5.626.2.1 `__ctype_type`

```
template<typename _CharT, typename _Traits>
typedef __ctype_type<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type
```

These are non-standard types.

Definition at line 87 of file `basic_ios.h`.

#### 5.626.2.2 \_\_num\_get\_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type
```

These are non-standard types.

Definition at line 91 of file basic\_ios.h.

#### 5.626.2.3 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type
```

These are non-standard types.

Definition at line 89 of file basic\_ios.h.

#### 5.626.2.4 char\_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_ios<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.




Definition at line 76 of file basic\_ios.h.

#### 5.626.2.5 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

##### Parameters

 <code>__e</code>	One of the members of the event enum.
 <code>__b</code>	Reference to the ios_base object.
 <code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 506 of file ios\_base.h.

#### 5.626.2.6 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 323 of file ios\_base.h.

#### 5.626.2.7 int\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::basic_ios< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 77 of file basic\_ios.h.

#### 5.626.2.8 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 398 of file ios\_base.h.

#### 5.626.2.9 off\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 79 of file basic\_ios.h.

#### 5.626.2.10 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 429 of file *ios\_base.h*.

#### 5.626.2.11 pos\_type

```
template<typename _CharT, typename _Traits>  
typedef _Traits::pos_type std::basic_ios<_CharT, _Traits>::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 78 of file *basic\_ios.h*.

#### 5.626.2.12 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK\_CUR* in the C standard library.
- *end*, equivalent to *SEEK\_END* in the C standard library.

Definition at line 461 of file *ios\_base.h*.



### 5.626.2.13 traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 80 of file basic\_ios.h.

## 5.626.3 Member Enumeration Documentation

### 5.626.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

Definition at line 489 of file ios\_base.h.

## 5.626.4 Constructor & Destructor Documentation

### 5.626.4.1 basic\_ios() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios (
    basic_streambuf< _CharT, _Traits > * __sb ) [inline], [explicit]
```

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 270 of file basic\_ios.h.

#### 5.626.4.2 ~basic\_ios()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ios<_CharT, _Traits>::~~basic_ios ( ) [inline], [virtual]
```

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

Definition at line 282 of file basic\_ios.h.

#### 5.626.4.3 basic\_ios() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::basic_ios ( ) [inline], [protected]
```

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 460 of file basic\_ios.h.

### 5.626.5 Member Function Documentation

#### 5.626.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get<\_CharT, \_Inlter>::do\_get(), std::money\_get<\_CharT, \_Inlter>::do\_get(), std::num\_get<\_CharT, \_Inlter>::do\_get(), std::time\_get<\_CharT, \_Inlter>::do\_get\_date(), std::time\_get<\_CharT, \_Inlter>::do\_get\_monthname(), std::time\_get<\_CharT, \_Inlter>::do\_get\_time(), std::time\_get<\_CharT, \_Inlter>::do\_get\_weekday(), std::time\_put<\_CharT, \_Outlter>::do\_put(), std::num\_put<\_CharT, \_Outlter>::do\_put(), std::time\_get<\_CharT, \_Inlter>::get(), and std::time\_put<\_CharT, \_Outlter>::put().

#### 5.626.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

#### 5.626.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit )
```

[Re]sets the error state.

##### Parameters

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

#### 5.626.5.4 copyfmt()

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs )
```

Copies fields of `__rhs` into this.

##### Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

**5.626.5.5 eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file `basic_ios.h`.

**5.626.5.6 exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`.

**5.626.5.7 exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except ) [inline]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 5.626.5.8 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, `std::basic_ios< char, _Traits >::operator!()`, and `std::regex_traits< _CharType >::value()`.

## 5.626.5.9 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline]
```

Retrieves the *empty* character.

## Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

## 5.626.5.10 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline]
```

Sets a new *empty* character.

## Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

## Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

## 5.626.5.11 flags() [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

## Returns

The format control flags for both input and output.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

## 5.626.5.12 flags() [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file `ios_base.h`.

**5.626.5.13 `getloc()`**

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, and `std::ws()`.

**5.626.5.14 `good()`**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.626.5.15 `imbue()`**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc )
```

Moves to a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

## 5.626.5.16 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

## 5.626.5.17 iword()

```
long& std::ios_base::iword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

## Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------



**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

**5.626.5.18 narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default ) const [inline]
```

Squeezes characters.

**Parameters**

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>↔

Definition at line 430 of file `basic_ios.h`.

**5.626.5.19 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

#### 5.626.5.20 operator!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

#### 5.626.5.21 precision() [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::operator<<()`.

#### 5.626.5.22 precision() [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

##### Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

##### Returns

The previous value of `precision()`.

Definition at line 700 of file `ios_base.h`.

**5.626.5.23 pword()**

```
void*& std::ios_base::pword (
    int __ix )    [inline], [inherited]
```

Access to void pointer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file ios\_base.h.

**5.626.5.24 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::rdbuf ( ) const    [inline]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file basic\_ios.h.

Referenced by std::ws().

**5.626.5.25 rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf (
    basic_streambuf<_CharT, _Traits > * __sb )
```

Changing the underlying buffer.

## Parameters

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

## 5.626.5.26 rdstate()

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

## 5.626.5.27 register\_callback()

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.626.5.28** `setf()` [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**5.626.5.29** `setf()` [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

#### 5.626.5.30 setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
    iostate __state ) [inline]
```

Sets additional flags in the error state.

##### Parameters

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::ws()`.

#### 5.626.5.31 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 5.626.5.32 tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**5.626.5.33 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline]
```

Ties this stream to an output stream.

**Parameters**

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**5.626.5.34 unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

## 5.626.5.35 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline]
```

Widens characters.

## Parameters

<code>__c</code>	The character to widen.
------------------	-------------------------

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

## 5.626.5.36 width() [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

## 5.626.5.37 width() [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.



**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

**5.626.5.38 `xalloc()`**

```
static int std::ios_base::xalloc ( ) throw ( )    [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iwor` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iwor` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iwor` and `pword` arrays.

**5.626.6 Member Data Documentation****5.626.6.1 `adjustfield`**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.626.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file ios\_base.h.

#### 5.626.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file ios\_base.h.

#### 5.626.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), and std::operator<<().

#### 5.626.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 381 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< char >::operator<<().

#### 5.626.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.626.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.626.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter >::do_get()`, and `std::num_put<_CharT, _OutIter >::do_put()`.

#### 5.626.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 5.626.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.

#### 5.626.6.11 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 5.626.6.12 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_Inlter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 5.626.6.13 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_Inlter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

#### 5.626.6.14 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

#### 5.626.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

#### 5.626.6.16 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

#### 5.626.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.626.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

#### 5.626.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file `ios_base.h`.

#### 5.626.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.626.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::operator<<()`.

#### 5.626.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 446 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

**5.626.6.23 right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

**5.626.6.24 scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

**5.626.6.25 showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put().

**5.626.6.26 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

**5.626.6.27 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

#### 5.626.6.28 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

#### 5.626.6.29 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 449 of file ios\_base.h.

#### 5.626.6.30 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

#### 5.626.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

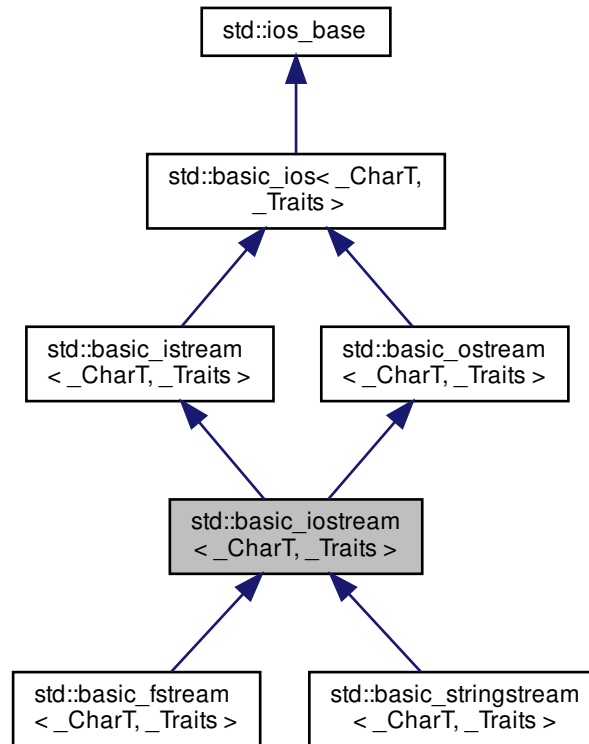
The documentation for this class was generated from the following files:

- [iosfwd](#)
- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)



## 5.627 std::basic\_iostream< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_iostream< \_CharT, \_Traits >:



### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event __e, ios_base & __b, int __i)`

- typedef \_ios\_Fmtflags [fmtflags](#)
  - typedef \_Traits::int\_type **int\_type**
  - typedef int **io\_state**
  - typedef \_ios\_istate [iostate](#)
  - typedef \_Traits::off\_type **off\_type**
  - typedef int **open\_mode**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef \_Traits::pos\_type **pos\_type**
  - typedef int **seek\_dir**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef [std::streamoff](#) **streamoff**
  - typedef [std::streampos](#) **streampos**
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_istream](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- virtual [~basic\\_istream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **\_M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT >  
[basic\\_ostream](#)< \_CharT, \_Traits > & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream](#)< char > & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- template<>  
[basic\\_istream](#)< wchar\_t > & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>  
[basic\\_istream](#)< char > & [ignore](#) ([streamsize](#) \_\_n)

- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (__streambuf_type *__sb)`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`

- [\\_\\_istream\\_type](#) & [operator>>](#) (short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned short &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned int &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long long &\_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long &\_\_n)

- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
- [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
- [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
- [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ()
- `int_type` [peek](#) ()
- [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
- [\\_\\_istream\\_type](#) & [unget](#) ()
- `int` [sync](#) ()
- `pos_type` [tellg](#) ()
- [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
- [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))

- [operator bool](#) () const
- `bool` [operator!](#) () const

- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
  - `__ostream_type & operator<< (unsigned long __n)`
  - `__ostream_type & operator<< (bool __n)`
  - `__ostream_type & operator<< (short __n)`
  - `__ostream_type & operator<< (unsigned short __n)`
  - `__ostream_type & operator<< (int __n)`
  - `__ostream_type & operator<< (unsigned int __n)`
  - `__ostream_type & operator<< (long long __n)`
  - `__ostream_type & operator<< (unsigned long long __n)`
- 
- `__ostream_type & operator<< (double __f)`
  - `__ostream_type & operator<< (float __f)`
  - `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

## Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

## Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

## Protected Member Functions

- **basic\_iostream** (const [basic\\_iostream](#) &)=delete
- **basic\_iostream** ([basic\\_iostream](#) &&\_\_rhs)
- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()

- `template<typename _ValueT >`  
`__ostream_type & _M_insert ( _ValueT __v)`
- `void _M_move (ios_base &) noexcept`
- `void _M_swap (ios_base & __rhs) noexcept`
- `void init (basic_streambuf< _CharT, _Traits > * __sb)`
- `void move (basic_ios & __rhs)`
- `void move (basic_ios && __rhs)`
- `basic_istream & operator= (const basic_istream &)=delete`
- `basic_istream & operator= (basic_istream && __rhs)`
- `void set_rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
- `void swap (basic_ostream & __rhs)`
- `void swap (basic_ios & __rhs) noexcept`
- `void swap (basic_istream & __rhs)`
- `void swap (basic_istream & __rhs)`

#### Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [ _S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

#### 5.627.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >
```

Template class `basic_istream`.

#### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 89 of file iosfwd.

## 5.627.2 Member Typedef Documentation

### 5.627.2.1 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.




Definition at line 89 of file basic\_ios.h.

### 5.627.2.2 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

 <i><b>__e</b></i>	One of the members of the event enum.
 <i><b>__b</b></i>	Reference to the ios_base object.
 <i><b>__i</b></i>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 506 of file ios\_base.h.

### 5.627.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:



- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 323 of file ios\_base.h.

#### 5.627.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 398 of file ios\_base.h.

### 5.627.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

### 5.627.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

## 5.627.3 Member Enumeration Documentation

### 5.627.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.627.4 Constructor & Destructor Documentation

### 5.627.4.1 `basic_istream()`

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
    basic_streambuf< _CharT, _Traits > * __sb ) [inline], [explicit]
```

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 849 of file istream.

### 5.627.4.2 `~basic_istream()`

```
template<typename _CharT, typename _Traits>
virtual std::basic_istream< _CharT, _Traits >::~~basic_istream ( ) [inline], [virtual]
```

Destructor does nothing.

Definition at line 856 of file istream.

## 5.627.5 Member Function Documentation

### 5.627.5.1 `_M_getloc()`

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 776 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

### 5.627.5.2 `_M_write()`

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

<code>_↵ _s</code>	The array to insert.
<code>_↵ _n</code>	Maximum number of characters to insert.

Definition at line 311 of file ostream.

**5.627.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**5.627.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >↵::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

#### 5.627.5.5 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of \_\_rhs into this.

##### Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

##### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

#### 5.627.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

#### 5.627.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.627.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 5.627.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, `std::basic_ios< char, _Traits >::operator!()`, and `std::regex_traits< _CharType >::value()`.

**5.627.5.10** `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**5.627.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**5.627.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**5.627.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file ios\_base.h.

**5.627.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.



**5.627.5.15 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

**5.627.5.16 get()** [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().


Definition at line 244 of file istream.tcc.

**5.627.5.17 get()** [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c ) [inherited]
```

Simple extraction.

**Parameters**

	The character in which to store data.
<b>__c</b>	

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file istream.tcc.

**5.627.5.18 get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Simple multiple-character extraction.

**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>__s</code> .
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Characters are extracted and stored into \_\_s until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

**5.627.5.19** `get()` [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>s</code> .

**Returns**

`*this`

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file `istream`.

**5.627.5.20** `get()` [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim ) [inherited]
```

Extraction into another streambuf.

**Parameters**

<code>__sb</code>	A streambuf in which to store data.
<code>__delim</code>	A "stop" character.

**Returns**

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF

- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

#### 5.627.5.21 get() [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb ) [inline], [inherited]
```

Extraction into another streambuf.

##### Parameters

<code>__sb</code>	A streambuf in which to store data.
-------------------	-------------------------------------

##### Returns

`*this`

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

#### 5.627.5.22 getline() [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

String extraction.

##### Parameters

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**5.627.5.23 `getline()`** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

**5.627.5.24** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

**5.627.5.25** `getloc()`

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, and `std::ws()`.

**5.627.5.26** `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.627.5.27** `ignore()` [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim ) [inherited]
```

Discarding characters.

**Parameters**

<code>__n</code>	Number of characters to discard.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**5.627.5.28 ignore()** [2/3]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.

## 5.627.5.29 ignore() [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

## 5.627.5.30 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

## 5.627.5.31 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```



All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

#### 5.627.5.32 `iword()`

```
long& std::ios_base::iword (  
    int __ix ) [inline], [inherited]
```

Access to integer array.

##### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

#### 5.627.5.33 `narrow()`

```
template<typename _CharT, typename _Traits>  
char std::basic_ios< _CharT, _Traits >::narrow (  
    char_type __c,  
    char __default ) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 430 of file `basic_ios.h`.

**5.627.5.34 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

**5.627.5.35 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 125 of file `basic_ios.h`.

**5.627.5.36 operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    __ostream_type &(*) (__ostream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**5.627.5.37 operator<<()** [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

**5.627.5.38 operator<<()** [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

**5.627.5.39 operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

## 5.627.5.40 operator&lt;&lt;() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 170 of file ostream.

## 5.627.5.41 operator&lt;&lt;() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    bool __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

**5.627.5.42** `operator<<()` [7/17]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n ) [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

## 5.627.5.43 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned short __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 181 of file `ostream`.

## 5.627.5.44 operator&lt;&lt;() [9/17]

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    int __n ) [inherited]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 106 of file `ostream.tcc`.

**5.627.5.45 operator<<() [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file `ostream`.

**5.627.5.46 operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 5.627.5.47 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

## 5.627.5.48 operator&lt;&lt;() [13/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.



**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

**5.627.5.49 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**5.627.5.50 operator<<() [15/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

$\leftarrow$	A variable of builtin floating point type.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 5.627.5.51 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p ) [inline], [inherited]
```

Pointer arithmetic inserters.

## Parameters

$\_ \leftarrow$	A variable of pointer type.
<i>p</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

## 5.627.5.52 operator&lt;&lt;() [17/17]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

**Parameters**

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

**5.627.5.53 operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(*) (__istream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

**5.627.5.54 operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

## 5.627.5.55 operator&gt;&gt;() [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

## 5.627.5.56 operator&gt;&gt;() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

## 5.627.5.57 operator&gt;&gt;() [5/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n ) [inherited]
```

Integer arithmetic extractors.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

**5.627.5.58 operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**5.627.5.59 operator>>() [7/17]**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n ) [inherited]
```

Integer arithmetic extractors.

**Parameters**

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**5.627.5.60 operator>>() [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits >::operator>> (
    unsigned int & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftrightarrow$ __n	A variable of builtin integral type.
--------------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**5.627.5.61 operator>>() [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits >::operator>> (
    long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftrightarrow$ __n	A variable of builtin integral type.
--------------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**5.627.5.62 operator>>() [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftrightarrow$ _n	A variable of builtin integral type.
-------------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**5.627.5.63 operator>>() [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftrightarrow$ _n	A variable of builtin integral type.
-------------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**5.627.5.64 operator>>() [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ __n	A variable of builtin integral type.
---------------------	--------------------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**5.627.5.65 operator>>() [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    float & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

$\leftarrow$ $\leftarrow$ $\leftarrow$ $\leftarrow$ <i>f</i>	A variable of builtin floating point type.
--	--



**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**5.627.5.66 operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**5.627.5.67 operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**5.627.5.68 operator>>() [16/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    void *& __p ) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

<code>__p</code>	A variable of pointer type.
------------------	-----------------------------

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**5.627.5.69 operator>>() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

**Parameters**

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, `failbit` is set.

Definition at line 212 of file `istream.tcc`.

#### 5.627.5.70 `peek()`

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

##### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is `false`, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

#### 5.627.5.71 `precision()` [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 5.627.5.72 `precision()` [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of precision().

Definition at line 700 of file ios\_base.h.

**5.627.5.73 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c ) [inherited]
```

Simple insertion.

**Parameters**

<code>__c</code>	The character to insert.
------------------	--------------------------

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

**5.627.5.74 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c ) [inherited]
```

Unextracting a single character.

**Parameters**

<code>_↔ _c</code>	The character to push back into the input stream.
------------------------	---

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**5.627.5.75 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

<code>_↔ _ix</code>	Index into the array.
-------------------------	-----------------------

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file `ios_base.h`.

**5.627.5.76** rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::rdbuf ( ) const [inline],
[inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file basic\_ios.h.

Referenced by std::ws().

**5.627.5.77** rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf (
    basic_streambuf<_CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**5.627.5.78 rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, ↵  
_Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< ↵  
char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

**5.627.5.79 read()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction without delimiters.

**Parameters**

<code>↵ __s</code>	A character array.
<code>↵ __n</code>	Maximum number of characters to store.

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

## 5.627.5.80 readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::readsome (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

## Returns

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

## 5.627.5.81 register\_callback()

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

## Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.



Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.627.5.82 `seekg()` [1/2]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos ) [inherited]
```

Changing the current read position.

##### Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

##### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

#### 5.627.5.83 `seekg()` [2/2]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current read position.

##### Parameters

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

#### 5.627.5.84 seekp() [1/2]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos ) [inherited]
```

Changing the current write position.

#### Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

#### 5.627.5.85 seekp() [2/2]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current write position.

#### Parameters

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

**5.627.5.86 setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**5.627.5.87 setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

**5.627.5.88 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

**5.627.5.89 sync()**

```
template<typename _CharT , typename _Traits >
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

**5.627.5.90 sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**5.627.5.91 tellg()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**5.627.5.92 tellp()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ( )
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

## 5.627.5.93 tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

## 5.627.5.94 tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (
    basic_ostream<_CharT, _Traits> * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

## 5.627.5.95 unget()

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**5.627.5.96 unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

**5.627.5.97 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

**Parameters**

<code>__c</code>	The character to widen.
------------------	-------------------------

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**5.627.5.98 width()** [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**5.627.5.99 width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.



Definition at line 723 of file ios\_base.h.

#### 5.627.5.100 write()

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n ) [inherited]
```

Character string insertion.

##### Parameters

<code>__s</code>	The array to insert.
<code>__n</code>	Maximum number of characters to insert.

##### Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

#### 5.627.5.101 xalloc()

```
static int std::ios_base::xalloc ( ) throw ( ) [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.627.6 Member Data Documentation

### 5.627.6.1 \_M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic\_istream<char>::get(), std::basic\_istream<char>::getline(), std::basic\_istream<char>::ignore(), std::basic\_istream<char>::peek(), std::basic\_istream<char>::putback(), std::basic\_istream<char>::read(), std::basic\_istream<char>::readsome(), and std::basic\_istream<char>::unget().

### 5.627.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of setf.

Definition at line 378 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_Outlter>::do\_put().

### 5.627.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file ios\_base.h.

### 5.627.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file ios\_base.h.

#### 5.627.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 5.627.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 381 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.627.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.627.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.627.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.627.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.627.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file `ios_base.h`.

#### 5.627.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**5.627.6.13 eofbit**

```
const iosstate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::eof()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_istream<char>::unget()`, and `std::ws()`.

**5.627.6.14 failbit**

```
const iosstate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::fail()`, `std::time_get<_CharT, _InIter>::get()`, and `std::basic_ostream<_CharT, _Traits>::sentry()`.

**5.627.6.15 fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file `ios_base.h`.

**5.627.6.16 floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

### 5.627.6.17 goodbit

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

### 5.627.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< char >::operator<<().

### 5.627.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for ifstream and fstream.

Definition at line 443 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

### 5.627.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 340 of file ios\_base.h.

**5.627.6.21 left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.627.6.22 oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::operator<<().

**5.627.6.23 out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for ofstream and fstream.

Definition at line 446 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos().

**5.627.6.24 right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.627.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.627.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

#### 5.627.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.627.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

#### 5.627.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.



### 5.627.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 449 of file `ios_base.h`.

### 5.627.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file `ios_base.h`.

### 5.627.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file `ios_base.h`.

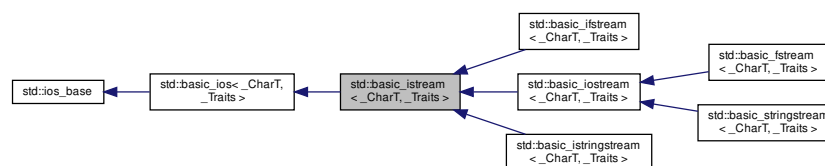
Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [istream](#)

## 5.628 std::basic\_istream<\_CharT, \_Traits> Class Template Reference

Inheritance diagram for `std::basic_istream<_CharT, _Traits>`:



## Classes

- class [sentry](#)

## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [basic\\_istream](#)< \_CharT, \_Traits > **\_\_istream\_type**
  - typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef \_CharT **char\_type**
  - enum [event](#) { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags **fmtflags**
  - typedef \_Traits::int\_type **int\_type**
  - typedef int **io\_state**
  - typedef \_ios\_istate **iostate**
  - typedef \_Traits::off\_type **off\_type**
  - typedef int **open\_mode**
  - typedef \_ios\_Openmode **openmode**
  - typedef \_Traits::pos\_type **pos\_type**
  - typedef int **seek\_dir**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef [std::streamoff](#) **streamoff**
  - typedef [std::streampos](#) **streampos**
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**

## Public Member Functions

- [basic\\_istream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- virtual [~basic\\_istream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **\_M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & **\_M\_getloc** () const
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool **bad** () const
- void **clear** ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & **copyfmt** (const [basic\\_ios](#) &\_\_rhs)
- bool **eof** () const
- [iostate](#) **exceptions** () const
- void **exceptions** ([iostate](#) \_\_except)
- bool **fail** () const
- [char\\_type](#) **fill** () const

- `char_type fill (char_type __ch)`
  - `fmtflags flags () const`
  - `fmtflags flags (fmtflags __fmtfl)`
  - `streamsize gcount () const`
  - `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `locale getloc () const`
  - `bool good () const`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __default) const`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

## Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`

- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

## Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`

- [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
- [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, ios\_base::seekdir)

- [operator bool](#) () const
- bool [operator!](#) () const

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- **basic\_istream** (const **basic\_istream** &)=delete
- **basic\_istream** (**basic\_istream** &&\_\_rhs)
- void **\_M\_cache\_locale** (const **locale** &\_\_loc)
- void **\_M\_call\_callbacks** (**event** \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  **\_\_istream\_type** & **\_M\_extract** (\_ValueT &\_\_v)
- **\_Words** & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** (**ios\_base** &) noexcept
- void **\_M\_swap** (**ios\_base** &\_\_rhs) noexcept
- void **init** (**basic\_streambuf**< \_CharT, \_Traits > \*\_\_sb)
- void **move** (**basic\_ios** &\_\_rhs)
- void **move** (**basic\_ios** &&\_\_rhs)
- **basic\_istream** & **operator=** (const **basic\_istream** &)=delete
- **basic\_istream** & **operator=** (**basic\_istream** &&\_\_rhs)
- void **set\_rdbuf** (**basic\_streambuf**< \_CharT, \_Traits > \*\_\_sb)
- void **swap** (**basic\_ios** &\_\_rhs) noexcept
- void **swap** (**basic\_istream** &\_\_rhs)

## Protected Attributes

- **\_Callback\_list** \* **\_M\_callbacks**
- const **\_\_ctype\_type** \* **\_M\_ctype**
- **iostate** **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- **fmtflags** **\_M\_flags**
- **streamsize** **\_M\_gcount**
- **locale** **\_M\_ios\_locale**
- **\_Words** **\_M\_local\_word** [**\_S\_local\_word\_size**]
- const **\_\_num\_get\_type** \* **\_M\_num\_get**
- const **\_\_num\_put\_type** \* **\_M\_num\_put**
- **streamsize** **\_M\_precision**
- **basic\_streambuf**< \_CharT, \_Traits > \* **\_M\_streambuf**
- **iostate** **\_M\_streambuf\_state**
- **basic\_ostream**< \_CharT, \_Traits > \* **\_M\_tie**
- **streamsize** **\_M\_width**
- **\_Words** \* **\_M\_word**
- int **\_M\_word\_size**
- **\_Words** **\_M\_word\_zero**

## Friends

- class **sentry**

### 5.628.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >
```

Template class basic\_istream.

#### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from basic\_streambuf to do the actual input.

Definition at line 83 of file iosfwd.

### 5.628.2 Member Typedef Documentation

#### 5.628.2.1 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file basic\_ios.h.

#### 5.628.2.2 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

<code>_e</code>	One of the members of the event enum.
<code>_b</code>	Reference to the ios_base object.
<code>_i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 506 of file ios\_base.h.

## 5.628.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 323 of file ios\_base.h.



#### 5.628.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- *badbit*
- *eofbit*
- *failbit*
- *goodbit*

Definition at line 398 of file *ios\_base.h*.

#### 5.628.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 429 of file *ios\_base.h*.

### 5.628.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

## 5.628.3 Member Enumeration Documentation

### 5.628.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.628.4 Constructor & Destructor Documentation

### 5.628.4.1 basic\_istream()

```
template<typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits >::basic_istream (  
    __streambuf_type * __sb ) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 93 of file `istream`.

#### 5.628.4.2 ~basic\_istream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_istream< _CharT, _Traits >::~~basic_istream ( ) [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 103 of file istream.

#### 5.628.5 Member Function Documentation

##### 5.628.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

##### 5.628.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

##### 5.628.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

## Parameters

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<char, _Traits>::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<char>::unset()`.

## 5.628.5.4 copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
    const basic_ios<_CharT, _Traits> & __rhs ) [inherited]
```

Copies fields of `__rhs` into this.

## Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

## Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 63 of file `basic_ios.tcc`.

## 5.628.5.5 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

### 5.628.5.6 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

### 5.628.5.7 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iosstate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

#### Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file basic\_ios.h.

#### 5.628.5.8 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator bool(), std::basic\_ios< char, \_Traits >::operator!(), and std::basic\_istream< \_CharType >::value().

#### 5.628.5.9 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::basic\_ios< char, \_Traits >::fill(), and std::operator<<().

#### 5.628.5.10 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

##### Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**5.628.5.11 flags()** [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**5.628.5.12 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file `ios_base.h`.

## 5.628.5.13 gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount ( ) const [inline]
```

Character counting.

## Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

## 5.628.5.14 get() [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (
    void )
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().


Definition at line 244 of file istream.tcc.

## 5.628.5.15 get() [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    char_type & __c )
```

Simple extraction.

## Parameters

 <code>__c</code>	The character in which to store data.
--	---------------------------------------



**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file `istream.tcc`.

**5.628.5.16 `get()`** [3/6]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim )
```

Simple multiple-character extraction.

**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>__s</code> .
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file `istream.tcc`.

## 5.628.5.17 get() [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::get (
    char_type * __s,
    streamsize __n ) [inline]
```

Simple multiple-character extraction.

## Parameters

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>s</code> .

## Returns

\*this

Returns `get(__s, __n, widen("\n"))`.

Definition at line 354 of file istream.

## 5.628.5.18 get() [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
    __streambuf_type & __sb,
    char_type __delim )
```

Extraction into another streambuf.

## Parameters

<code>__sb</code>	A streambuf in which to store data.
<code>__delim</code>	A "stop" character.

## Returns

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF

- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file `istream.tcc`.

#### 5.628.5.19 `get()` [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb ) [inline]
```

Extraction into another streambuf.

##### Parameters

<code>__sb</code>	A streambuf in which to store data.
-------------------	-------------------------------------

##### Returns

`*this`

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file `istream`.

#### 5.628.5.20 `getline()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim )
```

String extraction.

##### Parameters

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**5.628.5.21 getline()** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n ) [inline]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

**5.628.5.22** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim )
```

Explicit specialization declarations, defined in `src/istream.cc`.

**5.628.5.23** `getloc()`

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIt >::do_put()`, and `std::ws()`.

**5.628.5.24** `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.628.5.25** `ignore()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim )
```

Discarding characters.

## Parameters

<code>__n</code>	Number of characters to discard.
<code>__delim</code>	A "stop" character.

## Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

5.628.5.26 `ignore()` [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
    streamsize __n )
```

Simple extraction.

## Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.

**5.628.5.27 ignore()** [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void )
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**5.628.5.28 imbue()**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

**5.628.5.29 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::basic\_ios().

#### 5.628.5.30 iword()

```
long& std::ios_base::iword (  
    int __ix ) [inline], [inherited]
```

Access to integer array.

##### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

##### Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file ios\_base.h.

#### 5.628.5.31 narrow()

```
template<typename _CharT, typename _Traits>  
char std::basic_ios< _CharT, _Traits >::narrow (  
    char_type __c,  
    char __default ) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.



**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file `basic_ios.h`.

**5.628.5.32 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.628.5.33 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**5.628.5.34 operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(*) (__istream_type &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

## 5.628.5.35 operator&gt;&gt;() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    __ios_type &(*) (__ios_type &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

## 5.628.5.36 operator&gt;&gt;() [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    ios_base &(*) (ios_base &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

## 5.628.5.37 operator&gt;&gt;() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    bool & __n ) [inline]
```

Integer arithmetic extractors.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 168 of file `istream`.

**5.628.5.38 operator>>()** [5/17]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n )
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 122 of file `istream.tcc`.

**5.628.5.39 operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n ) [inline]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

## 5.628.5.40 operator&gt;&gt;() [7/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n )
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>__n</code>	A variable of builtin integral type.
----------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**5.628.5.41 `operator>>()` [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n ) [inline]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>__n</code>	A variable of builtin integral type.
----------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**5.628.5.42 `operator>>()` [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n ) [inline]
```

Integer arithmetic extractors.

## Parameters

$\_↵$	A variable of builtin integral type.
$\_n$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

## 5.628.5.43 operator&gt;&gt;() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n ) [inline]
```

Integer arithmetic extractors.

## Parameters

$\_↵$	A variable of builtin integral type.
$\_n$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

## 5.628.5.44 operator&gt;&gt;() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n ) [inline]
```

Integer arithmetic extractors.

**Parameters**

$\_↵$	A variable of builtin integral type.
$\_n$	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**5.628.5.45 `operator>>()` [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n ) [inline]
```

Integer arithmetic extractors.

**Parameters**

$\_↵$	A variable of builtin integral type.
$\_n$	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**5.628.5.46 `operator>>()` [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f ) [inline]
```

Floating point arithmetic extractors.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

## 5.628.5.47 operator&gt;&gt;() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f ) [inline]
```

Floating point arithmetic extractors.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

## 5.628.5.48 operator&gt;&gt;() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f ) [inline]
```

Floating point arithmetic extractors.



**Parameters**

$\leftarrow$	A variable of builtin floating point type.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$	
$f$	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**5.628.5.49 operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p ) [inline]
```

Basic arithmetic extractors.

**Parameters**

$\_ \leftarrow$	A variable of pointer type.
$\_p$	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**5.628.5.50 operator>>()** [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb )
```

Extracting into another streambuf.

## Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file istream.tcc.

## 5.628.5.51 peek()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void )
```

Looking ahead in the stream.

## Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file istream.tcc.

## 5.628.5.52 precision() [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

## 5.628.5.53 precision() [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of `precision()`.

Definition at line 700 of file `ios_base.h`.

**5.628.5.54 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c )
```

Unextracting a single character.

**Parameters**

<code>__c</code>	The character to push back into the input stream.
------------------	---

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**5.628.5.55 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

## Parameters

<code>_↵ _ix</code>	Index into the array.
-------------------------	-----------------------

## Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file ios\_base.h.

## 5.628.5.56 rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf ( ) const [inline],
[inherited]
```

Accessing the underlying buffer.

## Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file basic\_ios.h.

Referenced by std::ws().

## 5.628.5.57 rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
    basic_streambuf<_CharT, _Traits> * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**5.628.5.58 `rdbuf()`**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdbuf ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unset()`.

**5.628.5.59 `read()`**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n )
```

Extraction without delimiters.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

**5.628.5.60 readsome()**

```
template<typename _CharT , typename _Traits >
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n )
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called A here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the `streambuf`.

Definition at line 687 of file `istream.tcc`.

#### 5.628.5.61 `register_callback()`

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

##### Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.628.5.62 `seekg()` [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos )
```

Changing the current read position.

##### Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**5.628.5.63 seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
    off_type __off,
    ios_base::seekdir __dir )
```

Changing the current read position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**5.628.5.64 setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------



**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**5.628.5.65 setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

**5.628.5.66 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::ws().

#### 5.628.5.67 sync()

```
template<typename _CharT , typename _Traits >
int std::basic_istream< _CharT, _Traits >::sync (
    void )
```

Synchronizing the stream buffer.

##### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

##### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file istream.tcc.

#### 5.628.5.68 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**5.628.5.69 tellg()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void )
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**5.628.5.70 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**5.628.5.71 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 307 of file basic\_ios.h.

## 5.628.5.72 unget()

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
    void )
```

Unextracting the previous character.

## Returns

\*this

If rdbuf() is not null, calls rdbuf()->sungetc(c).

If rdbuf() is null or if sungetc() fails, sets badbit in the error state.

## Note

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

Definition at line 754 of file istream.tcc.

Referenced by std::\_\_detail::operator>>().

## 5.628.5.73 unsetf()

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

**5.628.5.74 `widen()`**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

**Parameters**

<code>__c</code>	The character to widen.
------------------	-------------------------

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**5.628.5.75 `width()` [1/2]**

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**5.628.5.76** width() [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of width().

Definition at line 723 of file ios\_base.h.

**5.628.5.77** xalloc()

```
static int std::ios_base::xalloc ( ) throw ( ) [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.628.6** Member Data Documentation**5.628.6.1** \_M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::_M_gcount [protected]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_istream<char>::peek()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsome()`, and `std::basic_istream<char>::unget()`.

#### 5.628.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outlter >::do_put()`.

#### 5.628.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

#### 5.628.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

#### 5.628.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 5.628.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 381 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _Inlter>::do_get()`, `std::num_put<_CharT, _Outlter>::do_put()`, and `std::basic_ostream<char>::operator<<()`.

#### 5.628.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.628.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.628.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.628.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.628.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.



**5.628.6.12 end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**5.628.6.13 eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::eof()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_istream< char >::unget()`, and `std::ws()`.

**5.628.6.14 failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry()`.

**5.628.6.15 fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file `ios_base.h`.

#### 5.628.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

#### 5.628.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<char>::flush()`, `std::basic_istream<char>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_ostream<char>::operator<<()`, `std::basic_istream<char>::operator>>()`, `std::operator>>()`, `std::basic_istream<char>::peek()`, `std::basic_ostream<char>::put()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsome()`, `std::basic_istream<char>::seekg()`, `std::basic_ostream<char>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char>::sync()`, and `std::basic_istream<char>::unget()`.

#### 5.628.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<char>::operator<<()`.

#### 5.628.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

**5.628.6.20 internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file `ios_base.h`.

**5.628.6.21 left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

**5.628.6.22 oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::operator<<()`.

**5.628.6.23 out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 446 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 5.628.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.628.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.628.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

#### 5.628.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.628.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

**5.628.6.29 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

**5.628.6.30 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 449 of file ios\_base.h.

**5.628.6.31 unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

**5.628.6.32 uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_Outlter >::do\_put().

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [istream](#)
- [istream.tcc](#)

## 5.629 std::basic\_istream&lt; \_CharT, \_Traits &gt;::sentry Class Reference

## Public Types

- typedef [\\_\\_istream\\_type](#)::[\\_\\_ctype\\_type](#) [\\_\\_ctype\\_type](#)
- typedef [\\_Traits](#)::[int\\_type](#) [\\_\\_int\\_type](#)
- typedef [basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_istream\\_type](#)
- typedef [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > [\\_\\_streambuf\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)

## Public Member Functions

- [sentry](#) ([basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > &[\\_\\_is](#), bool [\\_\\_noskipws](#)=false)
- [operator bool](#) () const

## 5.629.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >::sentry
```

Performs setup work for input streams.

Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 686 of file `istream`.

## 5.629.2 Member Typedef Documentation

## 5.629.2.1 traits\_type

```
template<typename _CharT, typename _Traits>
typedef \_Traits std::basic\_istream< \_CharT, \_Traits >::sentry::traits_type
```

Easy access to dependent types.

Definition at line 693 of file `istream`.

## 5.629.3 Constructor &amp; Destructor Documentation

## 5.629.3.1 sentry()

```
template<typename _CharT, typename _Traits>
std::basic\_istream< \_CharT, \_Traits >::sentry::sentry (
    basic\_istream< \_CharT, \_Traits > & \_\_is,
    bool \_\_noskipws = false ) [explicit]
```

The constructor performs all the work.

**Parameters**

<code>__is</code>	The input stream to guard.
<code>__noskipws</code>	Whether to consume whitespace or not.

If the stream state is good (`__is.good()` is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if `__noskipws` is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `istream.tcc`.

References `std::basic_ios<_CharT, _Traits>::good()`, and `std::ios_base::goodbit`.

**5.629.4 Member Function Documentation****5.629.4.1 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::sentry::operator bool ( ) const [inline], [explicit]
```

Quick status checking.

**Returns**

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

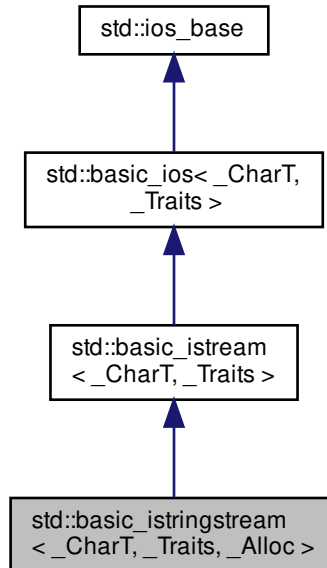
Definition at line 734 of file `istream`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 5.630 std::basic\_istream&lt; \_CharT, \_Traits, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event __e, ios_base & __b, int __i)`
- typedef `_ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_ios_istate iostate`
- typedef `traits_type::off_type off_type`
- typedef `int open_mode`
- typedef `_ios_Openmode openmode`
- typedef `traits_type::pos_type pos_type`



- typedef int **seek\_dir**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef [std::streamoff](#) **streamoff**
  - typedef [std::streampos](#) **streampos**
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >](#) [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_istream](#) ([ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_istream](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- **basic\_istream** (const [basic\\_istream](#) &)=delete
- **basic\_istream** ([basic\\_istream](#) &&\_\_rhs)
- [~basic\\_istream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & [M\\_getloc](#) () const
- void **M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream](#)< char > & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- template<>  
[basic\\_istream](#)< wchar\_t > & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>  
[basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- template<>  
[basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)

- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
  - [basic\\_istringstream](#) & [operator=](#) (const [basic\\_istringstream](#) &)=delete
  - [basic\\_istringstream](#) & [operator=](#) ([basic\\_istringstream](#) && \_\_rhs)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (void \*& \_\_p)
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
  - [\\_\\_stringbuf\\_type](#) \* [rdbuf](#) () const
  - [iosstate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iosstate](#) \_\_state)
  - [\\_\\_string\\_type](#) [str](#) () const
  - void [str](#) (const [\\_\\_string\\_type](#) & \_\_s)
  - void [swap](#) ([basic\\_istringstream](#) & \_\_rhs)
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - char\_type [widen](#) (char \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_istream\\_type](#) & (\*\_\_pf)(\_\_istream\_type &))
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([\\_\\_ios\\_type](#) & (\*\_\_pf)(\_\_ios\_type &))
  - [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) & (\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [\\_\\_istream\\_type](#) & [operator>>](#) (bool & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (short & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned short & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (int & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned int & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long long & \_\_n)
- [\\_\\_istream\\_type](#) & [operator>>](#) (unsigned long long & \_\_n)

- [\\_\\_istream\\_type](#) & [operator>>](#) (float &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (double &\_\_f)
- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ()
  - `int_type` [peek](#) ()
  - [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - `int` [sync](#) ()
  - `pos_type` [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - `bool` [operator!](#) () const

### Static Public Member Functions

- static `bool` [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static `int` [xalloc](#) () throw ()

## Static Public Attributes

- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iosstate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iosstate](#) eofbit
- static const [iosstate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iosstate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) & \_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
[\\_istream\\_type](#) & **\_M\_extract** (\_ValueT & \_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) & \_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void **move** ([basic\\_ios](#) & \_\_rhs)
- void **move** ([basic\\_ios](#) && \_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- void **swap** ([basic\\_ios](#) & \_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) & \_\_rhs)

## Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- `const __ctype_type` \* `_M_ctype`
- `iosstate` `_M_exception`
- `char_type` `_M_fill`
- `bool` `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- `const __num_get_type` \* `_M_num_get`
- `const __num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iosstate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- `int` `_M_word_size`
- `_Words` `_M_word_zero`

## 5.630.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_istream< _CharT, _Traits, _Alloc >
```

Controlling input for `std::string`.

## Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 100 of file `iosfwd`.

## 5.630.2 Member Typedef Documentation

### 5.630.2.1 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.




Definition at line 89 of file basic\_ios.h.

### 5.630.2.2 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

 <code>__e</code>	One of the members of the event enum.
 <code>__b</code>	Reference to the ios_base object.
 <code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 506 of file ios\_base.h.

### 5.630.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex

- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 323 of file ios\_base.h.

#### 5.630.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 398 of file ios\_base.h.

#### 5.630.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 429 of file *ios\_base.h*.

#### 5.630.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK\_CUR* in the C standard library.
- *end*, equivalent to *SEEK\_END* in the C standard library.

Definition at line 461 of file *ios\_base.h*.

### 5.630.3 Member Enumeration Documentation

#### 5.630.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

*erase\_event* is used during *~ios()* and *copyfmt()*. *imbue\_event* is used during *imbue()*. *copyfmt\_event* is used during *copyfmt()*.

Definition at line 489 of file *ios\_base.h*.



## 5.630.4 Constructor & Destructor Documentation

### 5.630.4.1 `basic_istream()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream (
    ios_base::openmode __mode = ios_base::in ) [inline], [explicit]
```

Default constructor starts with an empty string buffer.

#### Parameters

<code>__mode</code>	Whether the buffer can read, or write, or both.
---------------------	---

`ios_base::in` is automatically included in `__mode`.

Initializes `sb` using `__mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 417 of file `sstream`.

### 5.630.4.2 `basic_istream()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream (
    const __string_type & __str,
    ios_base::openmode __mode = ios_base::in ) [inline], [explicit]
```

Starts with an existing string buffer.

#### Parameters

<code>__str</code>	A string to copy as a starting buffer.
<code>__mode</code>	Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in `mode`.

Initializes `sb` using `str` and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 435 of file `sstream`.

## 5.630.4.3 ~basic\_istream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::~~basic_istream ( ) [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 446 of file sstream.

## 5.630.5 Member Function Documentation

## 5.630.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

## Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

## 5.630.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

## 5.630.5.3 clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

**5.630.5.4 copyfmt()**

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 63 of file `basic_ios.tcc`.

**5.630.5.5 eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

## 5.630.5.6 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
ios_base std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(ios\_base) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.630.5.7 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    ios_base __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file basic\_ios.h.

#### 5.630.5.8 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator bool(), std::basic\_ios< char, \_Traits >::operator!(), and std::basic\_ios< \_CharType >::value().

#### 5.630.5.9 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::basic\_ios< char, \_Traits >::fill(), and std::operator<<().

#### 5.630.5.10 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

##### Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

**5.630.5.11 flags()** [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**5.630.5.12 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file ios\_base.h.

#### 5.630.5.13 gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]
```

Character counting.

##### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

#### 5.630.5.14 get() [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

##### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

#### 5.630.5.15 get() [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c ) [inherited]
```

Simple extraction.

##### Parameters

<code>__c</code>	The character in which to store data.
------------------	---------------------------------------

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file istream.tcc.

**5.630.5.16 get()** [3/6]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Simple multiple-character extraction.

**Parameters**

__s	Pointer to an array.
__n	Maximum number of characters to store in __s.
__delim	A "stop" character.

**Returns**

\*this

Characters are extracted and stored into \_\_s until one of the following happens:

- \_\_n-1 characters are stored
- the input sequence reaches EOF
- the next character equals \_\_delim, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.



**5.630.5.17** `get()` [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>s</code> .

**Returns**

`*this`

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file `istream`.

**5.630.5.18** `get()` [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim ) [inherited]
```

Extraction into another streambuf.

**Parameters**

<code>__sb</code>	A streambuf in which to store data.
<code>__delim</code>	A "stop" character.

**Returns**

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF

- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, `failbit` is set in the stream's error state.

Definition at line 364 of file `istream.tcc`.

#### 5.630.5.19 `get()` [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb ) [inline], [inherited]
```

Extraction into another streambuf.

##### Parameters

<code>__sb</code>	A streambuf in which to store data.
-------------------	-------------------------------------

##### Returns

`*this`

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file `istream`.

#### 5.630.5.20 `getline()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

String extraction.

##### Parameters

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**5.630.5.21 `getline()`** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

**5.630.5.22** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

**5.630.5.23** `getloc()`

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, and `std::ws()`.

**5.630.5.24** `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.630.5.25** `ignore()` [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim ) [inherited]
```

Discarding characters.

**Parameters**

<code>__n</code>	Number of characters to discard.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**5.630.5.26 ignore()** [2/3]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n ) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.

## 5.630.5.27 ignore() [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

## 5.630.5.28 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

## 5.630.5.29 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

#### 5.630.5.30 `iword()`

```
long& std::ios_base::iword (  
    int __ix ) [inline], [inherited]
```

Access to integer array.

##### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

#### 5.630.5.31 `narrow()`

```
template<typename _CharT, typename _Traits>  
char std::basic_ios< _CharT, _Traits >::narrow (  
    char_type __c,  
    char __default ) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 430 of file `basic_ios.h`.

**5.630.5.32 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

**5.630.5.33 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 125 of file `basic_ios.h`.

**5.630.5.34 operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(*) (__istream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.



**5.630.5.35 operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

**5.630.5.36 operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

**5.630.5.37 operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

**Returns**

\*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 168 of file `istream`.

**5.630.5.38 operator>>()** [5/17]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    short & __n ) [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 122 of file `istream.tcc`.

**5.630.5.39 operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned short & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**5.630.5.40** `operator>>()` [7/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n ) [inherited]
```

Integer arithmetic extractors.

## Parameters

$\_↵$	A variable of builtin integral type.
$\_n$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

## 5.630.5.41 operator&gt;&gt;() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\_↵$	A variable of builtin integral type.
$\_n$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 182 of file `istream`.

## 5.630.5.42 operator&gt;&gt;() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 186 of file `istream`.

**5.630.5.43 `operator>>()` [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**5.630.5.44 `operator>>()` [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

## 5.630.5.45 operator&gt;&gt;() [12/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

## 5.630.5.46 operator&gt;&gt;() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    float & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**5.630.5.47 operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**5.630.5.48 operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin floating point type.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

## 5.630.5.49 operator&gt;&gt;() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p ) [inline], [inherited]
```

Basic arithmetic extractors.

## Parameters

$\_ \leftarrow$	A variable of pointer type.
$\_p$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

## 5.630.5.50 operator&gt;&gt;() [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.



#### Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

#### 5.630.5.51 `peek()`

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

#### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is `false`, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

#### 5.630.5.52 `precision()` [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 5.630.5.53 `precision()` [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

## Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

## Returns

The previous value of `precision()`.

Definition at line 700 of file `ios_base.h`.

## 5.630.5.54 putback()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c ) [inherited]
```

Unextracting a single character.

## Parameters

<code>__c</code>	The character to push back into the input stream.
------------------	---

## Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

## Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

## 5.630.5.55 pword()

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

<code>_ix</code>	Index into the array.
------------------	-----------------------

**Returns**

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file `ios_base.h`.

**5.630.5.56 `rdbuf()` [1/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**5.630.5.57 rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_istream< _CharT, _Traits, _Alloc >::rdbuf ( ) const [inline]
```

Accessing the underlying buffer.

**Returns**

The current basic\_stringbuf buffer.

This hides both signatures of std::basic\_ios::rdbuf().

Definition at line 486 of file sstream.

**5.630.5.58 rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See std::ios\_base::iosstate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 137 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

**5.630.5.59 read()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction without delimiters.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

**5.630.5.60 readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the `streambuf`'s buffer, `rdbuf()->in_avail()`, called A here:

- if  $A == -1$ , sets eofbit and extracts no characters
- if  $A == 0$ , extracts no characters
- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

#### 5.630.5.61 register\_callback()

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

##### Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.630.5.62 seekg() [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos ) [inherited]
```

Changing the current read position.

##### Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**5.630.5.63 seekg()** [2/2]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current read position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**5.630.5.64 setf()** [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>().

**5.630.5.65 setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file ios\_base.h.

**5.630.5.66 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.



Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::ws()`.

#### 5.630.5.67 `str()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istreamstream< _CharT, _Traits, _Alloc >::str ( ) const [inline]
```

Copying out the string buffer.

##### Returns

`rdbuf() -> str()`

Definition at line 494 of file `sstream`.

#### 5.630.5.68 `str()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istreamstream< _CharT, _Traits, _Alloc >::str (
    const __string_type & __s ) [inline]
```

Setting a new buffer.

##### Parameters

<code>__s</code>	The string to use as a new sequence.
------------------	--------------------------------------

Calls `rdbuf() -> str(s)`.

Definition at line 504 of file `sstream`.

#### 5.630.5.69 `sync()`

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

**5.630.5.70 sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**5.630.5.71 tellg()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**5.630.5.72 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**5.630.5.73 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 307 of file basic\_ios.h.

**5.630.5.74 unget()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If rdbuf() is not null, calls rdbuf()->sungetc(c).

If rdbuf() is null or if sungetc() fails, sets badbit in the error state.

**Note**

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

Definition at line 754 of file istream.tcc.

Referenced by std::\_\_detail::operator>>().

**5.630.5.75 unsetf()**

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

**Parameters**

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file ios\_base.h.

#### 5.630.5.76 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const    [inline], [inherited]
```

Widens characters.

##### Parameters

<code>__c</code>	The character to widen.
------------------	-------------------------

##### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file basic\_ios.h.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

#### 5.630.5.77 width() [1/2]

```
streamsize std::ios_base::width ( ) const    [inline], [inherited]
```

Flags access.

##### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _Outiter >::do_put()`.

#### 5.630.5.78 width() [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide )    [inline], [inherited]
```

Changing flags.

## Parameters

<code>__wide</code>	The new width value.
---------------------	----------------------

## Returns

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

5.630.5.79 `xalloc()`

```
static int std::ios_base::xalloc ( ) throw ( )    [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.630.6 Member Data Documentation

5.630.6.1 `_M_gcount`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount    [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, and `std::basic_istream< char >::unget()`.

#### 5.630.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.630.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

#### 5.630.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

#### 5.630.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 5.630.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 381 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<char>::operator<<()`.

#### 5.630.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.630.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.630.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.630.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### 5.630.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.



**5.630.6.12 end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.630.6.13 eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_Inlter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

**5.630.6.14 failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_get< \_CharT, \_Inlter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_Inlter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

**5.630.6.15 fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

## 5.630.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

## 5.630.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

## 5.630.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

## 5.630.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

#### 5.630.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file `ios_base.h`.

#### 5.630.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.630.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::operator<<()`.

#### 5.630.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 446 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 5.630.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.630.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.630.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

#### 5.630.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.630.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

**5.630.6.29 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file `ios_base.h`.

**5.630.6.30 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 449 of file `ios_base.h`.

**5.630.6.31 unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file `ios_base.h`.

**5.630.6.32 uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file `ios_base.h`.

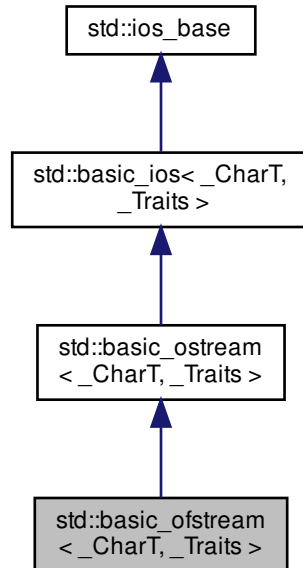
Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

## 5.631 std::basic\_ofstream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_ofstream< \_CharT, \_Traits >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback) (event __e, ios_base & __b, int __i)`
- typedef `_ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_ios_istate iostate`
- typedef `traits_type::off_type off_type`
- typedef `int open_mode`
- typedef `_ios_Openmode openmode`
- typedef `traits_type::pos_type pos_type`
- typedef `int seek_dir`
- typedef `_ios_Seekdir seekdir`

- typedef [std::streamoff](#) **streamoff**
  - typedef [std::streampos](#) **streampos**
  - typedef [\\_Traits](#) **traits\_type**
- 
- typedef [num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits>>](#) [\\_\\_num\\_get\\_type](#)

#### Public Member Functions

- [basic\\_ofstream](#) ()
- [basic\\_ofstream](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ofstream](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- **basic\_ofstream** (const [basic\\_ofstream](#) &)=delete
- **basic\_ofstream** ([basic\\_ofstream](#) &&\_\_rhs)
- [~basic\\_ofstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename [\\_ValueT](#) >  
[basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- bool [is\\_open](#) ()
- bool [is\\_open](#) () const
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- void [open](#) (const char \*\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- void [open](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \*\_\_p)
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- [basic\\_ofstream](#) & **operator=** (const [basic\\_ofstream](#) &)=delete
- [basic\\_ofstream](#) & **operator=** ([basic\\_ofstream](#) &&\_\_rhs)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)

- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `__filebuf_type * rdbuf () const`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `void swap (basic_ofstream & __rhs)`
- `pos_type tellp ()`
- `basic_ostream<_CharT, _Traits> * tie () const`
- `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tiestr)`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`

- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
  - `__ostream_type & operator<< (unsigned long __n)`
  - `__ostream_type & operator<< (bool __n)`
  - `__ostream_type & operator<< (short __n)`
  - `__ostream_type & operator<< (unsigned short __n)`
  - `__ostream_type & operator<< (int __n)`
  - `__ostream_type & operator<< (unsigned int __n)`
  - `__ostream_type & operator<< (long long __n)`
  - `__ostream_type & operator<< (unsigned long long __n)`
- 
- `__ostream_type & operator<< (double __f)`
  - `__ostream_type & operator<< (float __f)`
  - `__ostream_type & operator<< (long double __f)`



### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
  - `void _M_write (const char_type *__s, streamsize __n)`
  - `__ostream_type & write (const char_type *__s, streamsize __n)`
- 
- `operator bool () const`
  - `bool operator! () const`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Static Public Attributes

- `static const fmtflags adjustfield`
- `static const openmode app`
- `static const openmode ate`
- `static const iostate badbit`
- `static const fmtflags basefield`
- `static const seekdir beg`
- `static const openmode binary`
- `static const fmtflags boolalpha`
- `static const seekdir cur`
- `static const fmtflags dec`
- `static const seekdir end`
- `static const iostate eofbit`
- `static const iostate failbit`
- `static const fmtflags fixed`
- `static const fmtflags floatfield`
- `static const iostate goodbit`
- `static const fmtflags hex`
- `static const openmode in`
- `static const fmtflags internal`
- `static const fmtflags left`
- `static const fmtflags oct`
- `static const openmode out`
- `static const fmtflags right`
- `static const fmtflags scientific`
- `static const fmtflags showbase`
- `static const fmtflags showpoint`
- `static const fmtflags showpos`
- `static const fmtflags skipws`
- `static const openmode trunc`
- `static const fmtflags unitbuf`
- `static const fmtflags uppercase`

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 5.631.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ofstream< _CharT, _Traits >
```

Controlling output for files.

### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 701 of file `fstream`.

## 5.631.2 Member Typedef Documentation

### 5.631.2.1 `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

Definition at line 91 of file `basic_ios.h`.

### 5.631.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 506 of file `ios_base.h`.

### 5.631.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 323 of file `ios_base.h`.

### 5.631.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 398 of file `ios_base.h`.

#### 5.631.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

#### 5.631.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

### 5.631.3 Member Enumeration Documentation

#### 5.631.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.631.4 Constructor &amp; Destructor Documentation

## 5.631.4.1 basic\_ofstream() [1/3]

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream ( ) [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 727 of file `fstream`.

## 5.631.4.2 basic\_ofstream() [2/3]

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
    const char * __s,
    ios_base::openmode __mode = ios_base::out ) [inline], [explicit]
```

Create an output file stream.

## Parameters

<code>__s</code>	Null terminated string specifying the filename.
<code>__mode</code>	Open file in specified mode (see <code>std::ios_base</code> ).

`ios_base::out` is automatically included in `__mode`.

Definition at line 738 of file `fstream`.

## 5.631.4.3 basic\_ofstream() [3/3]

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::out ) [inline], [explicit]
```

Create an output file stream.

#### Parameters

<code>__s</code>	std::string specifying the filename.
<code>__mode</code>	Open file in specified mode (see std::ios_base).

ios\_base::out is automatically included in `__mode`.

Definition at line 755 of file fstream.

#### 5.631.4.4 ~basic\_ofstream()

```
template<typename _CharT, typename _Traits >
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream ( ) [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 792 of file fstream.

### 5.631.5 Member Function Documentation

#### 5.631.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

#### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

#### 5.631.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

<code>_↵ _s</code>	The array to insert.
<code>_↵ _n</code>	Maximum number of characters to insert.

Definition at line 311 of file ostream.

**5.631.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**5.631.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >↵::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().



#### 5.631.5.5 close()

```
template<typename _CharT, typename _Traits >
void std::basic_ofstream< _CharT, _Traits >::close ( ) [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 904 of file `fstream`.

#### 5.631.5.6 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of `__rhs` into this.

##### Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

##### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

#### 5.631.5.7 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

## 5.631.5.8 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.631.5.9 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file basic\_ios.h.

### 5.631.5.10 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator bool(), std::basic\_ios< char, \_Traits >::operator!(), and std::basic\_ios< \_CharType >::value().

### 5.631.5.11 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::basic\_ios< char, \_Traits >::fill(), and std::operator<<().

### 5.631.5.12 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

#### Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

**5.631.5.13 flags()** [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**5.631.5.14 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file ios\_base.h.

#### 5.631.5.15 flush()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( ) [inherited]
```

Synchronizing the stream buffer.

##### Returns

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

#### 5.631.5.16 getloc()

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, and `std::ws()`.

#### 5.631.5.17 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 5.631.5.18 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

## 5.631.5.19 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

## 5.631.5.20 is\_open()

```
template<typename _CharT , typename _Traits >
bool std::basic_ofstream< _CharT, _Traits >::is_open ( ) [inline]
```

Wrapper to test for an open file.

## Returns

`rdbuf()->is_open()`

Definition at line 833 of file `fstream`.

## 5.631.5.21 iword()

```
long& std::ios_base::iword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

**Parameters**

<code><a href="#">↵</a> _ix</code>	Index into the array.
--	-----------------------

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

**5.631.5.22 narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↵html>

Definition at line 430 of file `basic_ios.h`.

**5.631.5.23 open()** [1/2]

```
template<typename _CharT , typename _Traits >
void std::basic_ofstream< _CharT, _Traits >::open (
    const char * __s,
    ios_base::openmode __mode = ios_base::out ) [inline]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 851 of file `fstream`.

**5.631.5.24 open()** [2/2]

```
template<typename _CharT , typename _Traits >
void std::basic_ofstream< _CharT, _Traits >::open (
    const std::string & __s,
    ios_base::openmode __mode = ios_base::out ) [inline]
```

Opens an external file.

**Parameters**

<code>__s</code>	The name of the file.
<code>__mode</code>	The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 871 of file `fstream`.

**5.631.5.25 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.



**5.631.5.26 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**5.631.5.27 operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(*) (__ostream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**5.631.5.28 operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**5.631.5.29 operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

**5.631.5.30 operator<<() [4/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

## 5.631.5.31 operator&lt;&lt;() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
    unsigned long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 170 of file ostream.

## 5.631.5.32 operator&lt;&lt;() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
    bool __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 174 of file `ostream`.

**5.631.5.33 operator<<() [7/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n ) [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

**5.631.5.34 operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

## 5.631.5.35 operator&lt;&lt;() [9/17]

```
template<typename _CharT, typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::operator<< (
    int __n ) [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 106 of file ostream.tcc.

## 5.631.5.36 operator&lt;&lt;() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
    unsigned int __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 192 of file `ostream`.

**5.631.5.37 operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n )    [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

**5.631.5.38 operator<<() [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n )    [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

## 5.631.5.39 operator&lt;&lt;() [13/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
    double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

<code>__f</code>	A variable of builtin floating point type.
------------------	--

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

## 5.631.5.40 operator&lt;&lt;() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
    float __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**5.631.5.41 operator<<() [15/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**5.631.5.42 operator<<() [16/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p ) [inline], [inherited]
```

Pointer arithmetic inserters.

## Parameters

<code>_p</code>	A variable of pointer type.
-----------------	-----------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

5.631.5.43 `operator<<()` [17/17]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

## Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets `failbit` in the error state

If the function inserts no characters, `failbit` is set.

Definition at line 120 of file `ostream.tcc`.



**5.631.5.44** `precision()` [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.631.5.45** `precision()` [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of `precision()`.

Definition at line 700 of file `ios_base.h`.

**5.631.5.46** `put()`

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c ) [inherited]
```

Simple insertion.

**Parameters**

<code>__c</code>	The character to insert.
------------------	--------------------------

**Returns**

\*this

Tries to insert \_\_c.

**Note**

This function is not overloaded on signed char and unsigned char.


Definition at line 149 of file ostream.tcc.

**5.631.5.47 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

 __ix	Index into the array.
---	-----------------------

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file ios\_base.h.

**5.631.5.48 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**5.631.5.49 `rdbuf()` [2/2]**

```
template<typename _CharT, typename _Traits >
__filebuf_type* std::basic_ofstream< _CharT, _Traits >::rdbuf ( ) const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 825 of file `fstream`.

**5.631.5.50 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unset()`.

**5.631.5.51 `register_callback()`**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

## Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

## 5.631.5.52 seekp() [1/2]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::seekp (
    pos_type __pos ) [inherited]
```

Changing the current write position.

## Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

## 5.631.5.53 seekp() [2/2]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current write position.

## Parameters

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

#### 5.631.5.54 `setf()` [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

##### Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

#### 5.631.5.55 `setf()` [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

##### Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>fmtfl</code> .

##### Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file ios\_base.h.

#### 5.631.5.56 setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

##### Parameters

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See std::ios\_base::iostate for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::ws().

#### 5.631.5.57 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 5.631.5.58 tellp()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ( )
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**5.631.5.59 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**5.631.5.60 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

## 5.631.5.61 unsetf()

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

## Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

## 5.631.5.62 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

## Parameters

<code>__c</code>	The character to widen.
------------------	-------------------------

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.



**5.631.5.63** `width()` [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**5.631.5.64** `width()` [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

**5.631.5.65** `write()`

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n ) [inherited]
```

Character string insertion.

**Parameters**

<code>__s</code>	The array to insert.
<code>__n</code>	Maximum number of characters to insert.

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

**5.631.5.66 xalloc()**

```
static int std::ios_base::xalloc ( ) throw ( )    [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.631.6 Member Data Documentation**

#### 5.631.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.631.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

#### 5.631.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

#### 5.631.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 5.631.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 381 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.631.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.631.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.631.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.631.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### 5.631.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.

**5.631.6.11 end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.631.6.12 eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

**5.631.6.13 failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

**5.631.6.14 fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

## 5.631.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

## 5.631.6.16 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

## 5.631.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

## 5.631.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

#### 5.631.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file `ios_base.h`.

#### 5.631.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.631.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::operator<<()`.

#### 5.631.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 446 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 5.631.6.23 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.631.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.631.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

#### 5.631.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.631.6.27 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.



**5.631.6.28 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

**5.631.6.29 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 449 of file ios\_base.h.

**5.631.6.30 unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

**5.631.6.31 uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file ios\_base.h.

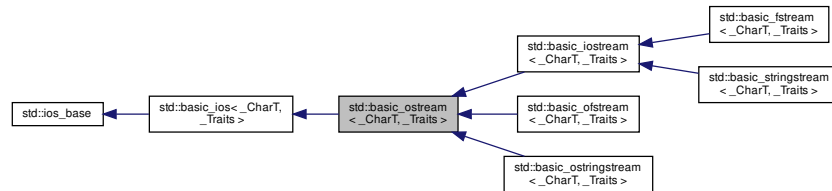
Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

- [fstream](#)

## 5.632 std::basic\_ostream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_ostream< \_CharT, \_Traits >:



## Classes

- class [sentry](#)

## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
- typedef [basic\\_ostream](#)< \_CharT, \_Traits > **\_\_ostream\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags **fmtflags**
- typedef \_Traits::int\_type **int\_type**
- typedef int **io\_state**
- typedef \_ios\_istate **istate**
- typedef \_Traits::off\_type **off\_type**
- typedef int **open\_mode**
- typedef \_ios\_Openmode **openmode**
- typedef \_Traits::pos\_type **pos\_type**
- typedef int **seek\_dir**
- typedef \_ios\_Seekdir **seekdir**
- typedef [std::streamoff](#) **streamoff**
- typedef [std::streampos](#) **streampos**
- typedef \_Traits **traits\_type**

- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**

## Public Member Functions

- [basic\\_ostream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - virtual [~basic\\_ostream](#) ()
  - const [locale](#) & [\\_M\\_getloc](#) () const
  - [template](#)<[typename](#) [\\_ValueT](#) >  
[basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
  - void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
  - bool [bad](#) () const
  - void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
  - [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
  - bool [eof](#) () const
  - [iostate](#) [exceptions](#) () const
  - void [exceptions](#) ([iostate](#) \_\_except)
  - bool [fail](#) () const
  - [char\\_type](#) [fill](#) () const
  - [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
  - [fmtflags](#) [flags](#) () const
  - [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
  - [\\_\\_ostream\\_type](#) & [flush](#) ()
  - [locale](#) [getloc](#) () const
  - bool [good](#) () const
  - [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
  - long & [iword](#) (int \_\_ix)
  - [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \*\_\_p)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
  - [iostate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [\\_\\_ostream\\_type](#) & [seekp](#) ([pos\\_type](#))
  - [\\_\\_ostream\\_type](#) & [seekp](#) ([off\\_type](#), [ios\\_base::seekdir](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [pos\\_type](#) [tellp](#) ()
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type](#) [widen](#) ([char](#) \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))

- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
  - void [\\_M\\_write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- 
- [operator bool](#) () const
  - bool [operator!](#) () const

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags adjustfield](#)
- static const [openmode app](#)
- static const [openmode ate](#)
- static const [iostate badbit](#)
- static const [fmtflags basefield](#)
- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- **basic\_ostream** ([basic\\_iostream](#)< [\\_CharT](#), [\\_Traits](#) > &)
- **basic\_ostream** (const [basic\\_ostream](#) &)=delete
- **basic\_ostream** ([basic\\_ostream](#) &&\_\_rhs)
- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()

- template<typename \_ValueT >  
    \_\_ostream\_type & **M\_insert** (\_ValueT \_\_v)
- void **M\_move** (ios\_base &) noexcept
- void **M\_swap** (ios\_base & \_\_rhs) noexcept
- void **init** (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
- void **move** (basic\_ios & \_\_rhs)
- void **move** (basic\_ios && \_\_rhs)
- basic\_ostream & **operator=** (const basic\_ostream &)=delete
- basic\_ostream & **operator=** (basic\_ostream && \_\_rhs)
- void **set\_rdbuf** (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
- void **swap** (basic\_ostream & \_\_rhs)
- void **swap** (basic\_ios & \_\_rhs) noexcept

#### Protected Attributes

- \_Callback\_list \* **M\_callbacks**
- const \_\_ctype\_type \* **M\_ctype**
- iostate **M\_exception**
- char\_type **M\_fill**
- bool **M\_fill\_init**
- fmtflags **M\_flags**
- locale **M\_ios\_locale**
- \_Words **M\_local\_word** [\_S\_local\_word\_size]
- const \_\_num\_get\_type \* **M\_num\_get**
- const \_\_num\_put\_type \* **M\_num\_put**
- streamsize **M\_precision**
- basic\_streambuf< \_CharT, \_Traits > \* **M\_streambuf**
- iostate **M\_streambuf\_state**
- basic\_ostream< \_CharT, \_Traits > \* **M\_tie**
- streamsize **M\_width**
- \_Words \* **M\_word**
- int **M\_word\_size**
- \_Words **M\_word\_zero**

#### Friends

- class **sentry**

#### 5.632.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >
```

Template class basic\_ostream.

**Template Parameters**

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

Definition at line 86 of file `iosfwd`.

**5.632.2 Member Typedef Documentation****5.632.2.1 `__num_get_type`**

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

Definition at line 91 of file `basic_ios.h`.

**5.632.2.2 `event_callback`**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

**Parameters**

<code>↵ __e</code>	One of the members of the event enum.
<code>↵ __b</code>	Reference to the <code>ios_base</code> object.
<code>↵ __i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 506 of file `ios_base.h`.

### 5.632.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- *boolalpha*
- *dec*
- *fixed*
- *hex*
- *internal*
- *left*
- *oct*
- *right*
- *scientific*
- *showbase*
- *showpoint*
- *showpos*
- *skipws*
- *unitbuf*
- *uppercase*
- *adjustfield*
- *basefield*
- *floatfield*

Definition at line 323 of file *ios\_base.h*.

### 5.632.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- *badbit*
- *eofbit*
- *failbit*
- *goodbit*

Definition at line 398 of file *ios\_base.h*.



#### 5.632.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

#### 5.632.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

### 5.632.3 Member Enumeration Documentation

#### 5.632.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.632.4 Constructor &amp; Destructor Documentation

## 5.632.4.1 basic\_ostream()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::basic_ostream (
    __streambuf_type * __sb ) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 84 of file ostream.

## 5.632.4.2 ~basic\_ostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream ( ) [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 93 of file ostream.

## 5.632.5 Member Function Documentation

## 5.632.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

## 5.632.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
    const char_type * __s,
    streamsize __n ) [inline]
```

Core write functionality, without sentry.

**Parameters**

<code>_↵ _s</code>	The array to insert.
<code>_↵ _n</code>	Maximum number of characters to insert.

Definition at line 311 of file ostream.

**5.632.5.3 bad()**

```
template<typename _CharT, typename _Traits>  
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**5.632.5.4 clear()**

```
template<typename _CharT , typename _Traits >  
void std::basic_ios< _CharT, _Traits >::clear (   
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< char >↵  
::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

## 5.632.5.5 copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of \_\_rhs into this.

## Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

## Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

## 5.632.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

## 5.632.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.632.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 5.632.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, `std::basic_ios< char, _Traits >::operator!()`, and `std::regex_traits< _CharType >::value()`.

**5.632.5.10 fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ostream< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file basic\_ostream.h.

Referenced by std::basic\_ostream< char, \_Traits >::copyfmt(), std::basic\_ostream< char, \_Traits >::fill(), and std::operator<<().

**5.632.5.11 fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ostream< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ostream.h.

**5.632.5.12 flags()** [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**5.632.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file `ios_base.h`.

**5.632.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( )
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file `ostream.tcc`.

## 5.632.5.15 getloc()

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, and `std::ws()`.

## 5.632.5.16 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

## 5.632.5.17 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------



**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**5.632.5.18 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**5.632.5.19 iword()**

```
long& std::ios_base::iword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file ios\_base.h.

#### 5.632.5.20 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 430 of file basic\_ios.h.

#### 5.632.5.21 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file basic\_ios.h.

**5.632.5.22 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**5.632.5.23 operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(*) (__ostream_type &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**5.632.5.24 operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ios_type &(*) (__ios_type &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**5.632.5.25 operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(*) (ios_base &) __pf ) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

**5.632.5.26 operator<<() [4/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n ) [inline]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

## 5.632.5.27 operator&lt;&lt;() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n ) [inline]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 170 of file ostream.

## 5.632.5.28 operator&lt;&lt;() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    bool __n ) [inline]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 174 of file `ostream`.

**5.632.5.29 operator<<() [7/17]**

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n )
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

**5.632.5.30 operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n ) [inline]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

## 5.632.5.31 operator&lt;&lt;() [9/17]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    int __n )
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 106 of file ostream.tcc.

## 5.632.5.32 operator&lt;&lt;() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n ) [inline]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 192 of file `ostream`.

**5.632.5.33 `operator<<()` [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n ) [inline]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

**5.632.5.34 `operator<<()` [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n ) [inline]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

## 5.632.5.35 operator&lt;&lt;() [13/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f ) [inline]
```

Floating point arithmetic inserters.

## Parameters

<code>__f</code>	A variable of builtin floating point type.
------------------	--

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

## 5.632.5.36 operator&lt;&lt;() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f ) [inline]
```

Floating point arithmetic inserters.



**Parameters**

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**5.632.5.37 operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f ) [inline]
```

Floating point arithmetic inserters.

**Parameters**

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**5.632.5.38 operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p ) [inline]
```

Pointer arithmetic inserters.

## Parameters

<code>_p</code>	A variable of pointer type.
-----------------	-----------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

## 5.632.5.39 operator&lt;&lt;() [17/17]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb )
```

Extracting from another streambuf.

## Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

**5.632.5.40 precision()** [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), and std::operator<<().

**5.632.5.41 precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of precision().

Definition at line 700 of file ios\_base.h.

**5.632.5.42 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c )
```

Simple insertion.

**Parameters**

<code>__c</code>	The character to insert.
------------------	--------------------------

**Returns**

\*this

Tries to insert \_\_c.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

**5.632.5.43 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file ios\_base.h.

**5.632.5.44 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::rdbuf ( ) const [inline],
[inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file `basic_ios.h`.

Referenced by `std::ws()`.

**5.632.5.45 `rdbuf()`** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

**Parameters**

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**5.632.5.46 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See std::ios\_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 137 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

**5.632.5.47 register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback \_\_fn with parameter \_\_index.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.632.5.48 seekp()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos )
```

Changing the current write position.

**Parameters**

<code>__pos</code>	A file position object.
--------------------	-------------------------

**Returns**

\*this

If fail() is not true, calls rdbuf()->pubseekpos(pos). If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

**5.632.5.49** `seekp()` [2/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir )
```

Changing the current write position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

**5.632.5.50** `setf()` [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**5.632.5.51** `setf()` [2/2]

```

fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]

```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>fmtfl</code> .

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

**5.632.5.52** `setstate()`

```

template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]

```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

**5.632.5.53** `sync_with_stdio()`

```

static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]

```

Interaction with the standard C I/O objects.



#### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 5.632.5.54 `tellp()`

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ( )
```

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

#### 5.632.5.55 `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

#### 5.632.5.56 `tie()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 307 of file basic\_ios.h.

## 5.632.5.57 unsetf()

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

## Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file ios\_base.h.

## 5.632.5.58 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

## Parameters

<code>__c</code>	The character to widen.
<code>__C</code>	

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

#### 5.632.5.59 `width()` [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.632.5.60 `width()` [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

#### Parameters

<code>__wide</code>	The new width value.
---------------------	----------------------

#### Returns

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

## 5.632.5.61 write()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n )
```

Character string insertion.

## Parameters

<code>__s</code>	The array to insert.
<code>__n</code>	Maximum number of characters to insert.

## Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

## 5.632.5.62 xalloc()

```
static int std::ios_base::xalloc ( ) throw ( )    [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.632.6 Member Data Documentation

### 5.632.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

### 5.632.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

### 5.632.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

### 5.632.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 5.632.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 381 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<char>::operator<<()`.

#### 5.632.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file `ios_base.h`.

#### 5.632.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file `ios_base.h`.

#### 5.632.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.632.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.632.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.

#### 5.632.6.11 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.632.6.12 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::eof()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_istream<char>::unget()`, and `std::ws()`.

## 5.632.6.13 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

## 5.632.6.14 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

## 5.632.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 384 of file ios\_base.h.

## 5.632.6.16 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_↵get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_↵get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_↵ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std\_↵::basic\_istream< char >::unget().



### 5.632.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

### 5.632.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file ios\_base.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

### 5.632.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file ios\_base.h.

### 5.632.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.632.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::operator<<().

#### 5.632.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for ofstream and fstream.

Definition at line 446 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos().

#### 5.632.6.23 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.632.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.632.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.632.6.26 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

**5.632.6.27 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

**5.632.6.28 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

**5.632.6.29 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 449 of file ios\_base.h.

**5.632.6.30 unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

## 5.632.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_Outlter >::do\_put().

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [ostream](#)
- [ostream.tcc](#)

## 5.633 std::basic\_ostream&lt; \_CharT, \_Traits &gt;::sentry Class Reference

## Public Member Functions

- [sentry](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)
- [~sentry](#) ()
- [operator bool](#) () const

## 5.633.1 Detailed Description

```
template<typename _CharT, typename _Traits>  
class std::basic_ostream< _CharT, _Traits >::sentry
```

Performs setup work for output streams.

Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 426 of file ostream.

## 5.633.2 Constructor &amp; Destructor Documentation

## 5.633.2.1 sentry()

```
template<typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits >::sentry::sentry (  
    basic_ostream< _CharT, _Traits > &__os ) [explicit]
```

The constructor performs preparatory work.

#### Parameters

<code>__os</code>	The output stream to guard.
-------------------	-----------------------------

If the stream state is good (`__os.good()` is true), then if the stream is tied to another output stream, `is.tie()->flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `ostream.tcc`.

References `std::ios_base::failbit`, `std::basic_ios<_CharT, _Traits>::good()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_ios<_CharT, _Traits>::tie()`.

#### 5.633.2.2 `~sentry()`

```
template<typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits>::sentry::~sentry ( ) [inline]
```

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 456 of file `ostream`.

#### 5.633.3 Member Function Documentation

##### 5.633.3.1 `operator bool()`

```
template<typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits>::sentry::operator bool ( ) const [inline], [explicit]
```

Quick status checking.

#### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

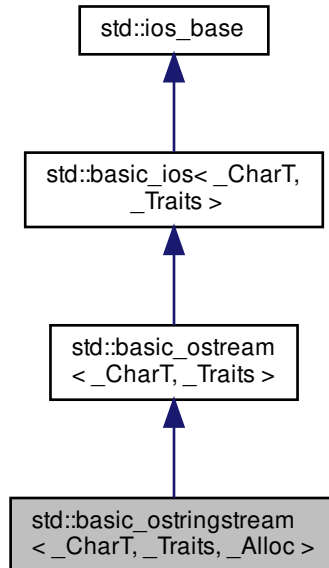
Definition at line 478 of file `ostream`.

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

## 5.634 std::basic\_ostringstream&lt; \_CharT, \_Traits, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event __e, ios_base & __b, int __i)`
- typedef `_ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_ios_istate iostate`
- typedef `traits_type::off_type off_type`
- typedef `int open_mode`
- typedef `_ios_Openmode openmode`
- typedef `traits_type::pos_type pos_type`

- typedef int **seek\_dir**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef **std::streamoff** **streamoff**
  - typedef **std::streampos** **streampos**
  - typedef \_Traits **traits\_type**
- 
- typedef **num\_get**< \_CharT, **istreambuf\_iterator**< \_CharT, \_Traits > > **\_\_num\_get\_type**

#### Public Member Functions

- **basic\_ostringstream** (**ios\_base::openmode** \_\_mode=**ios\_base::out**)
- **basic\_ostringstream** (const **\_\_string\_type** &\_\_str, **ios\_base::openmode** \_\_mode=**ios\_base::out**)
- **basic\_ostringstream** (const **basic\_ostringstream** &)=delete
- **basic\_ostringstream** (**basic\_ostringstream** &&\_\_rhs)
- **~basic\_ostringstream** ()
- const **locale** & **\_M\_getloc** () const
- template<typename \_ValueT >  
**basic\_ostream**< \_CharT, \_Traits > & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_setstate** (**iostate** \_\_state)
- bool **bad** () const
- void **clear** (**iostate** \_\_state=**goodbit**)
- **basic\_ios** & **copyfmt** (const **basic\_ios** &\_\_rhs)
- bool **eof** () const
- **iostate** **exceptions** () const
- void **exceptions** (**iostate** \_\_except)
- bool **fail** () const
- char\_type **fill** () const
- char\_type **fill** (char\_type \_\_ch)
- **fmtflags** **flags** () const
- **fmtflags** **flags** (**fmtflags** \_\_fmtfl)
- **\_\_ostream\_type** & **flush** ()
- **locale** **getloc** () const
- bool **good** () const
- **locale** **imbue** (const **locale** &\_\_loc)
- long & **word** (int \_\_ix)
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- **\_\_ostream\_type** & **operator<<** (const void \*\_\_p)
- **\_\_ostream\_type** & **operator<<** (**\_\_streambuf\_type** \*\_\_sb)
- **basic\_ostringstream** & **operator=** (const **basic\_ostringstream** &)=delete
- **basic\_ostringstream** & **operator=** (**basic\_ostringstream** &&\_\_rhs)
- **streamsize** **precision** () const
- **streamsize** **precision** (**streamsize** \_\_prec)
- void \*& **pword** (int \_\_ix)
- **basic\_streambuf**< \_CharT, \_Traits > \* **rdbuf** (**basic\_streambuf**< \_CharT, \_Traits > \*\_\_sb)
- **\_\_stringbuf\_type** \* **rdbuf** () const
- **iostate** **rdstate** () const
- void **register\_callback** (**event\_callback** \_\_fn, int \_\_index)

- [\\_\\_ostream\\_type](#) & [seekp](#) (pos\_type)
- [\\_\\_ostream\\_type](#) & [seekp](#) (off\_type, ios\_base::seekdir)
- [fmtflags](#) [setf](#) (fmtflags \_\_fmtfl)
- [fmtflags](#) [setf](#) (fmtflags \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) (iostate \_\_state)
- [\\_\\_string\\_type](#) [str](#) () const
- void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
- void [swap](#) (basic\_ostream &\_\_rhs)
- pos\_type [tellp](#) ()
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) () const
- [basic\\_ostream](#)<\_CharT, \_Traits> \* [tie](#) ([basic\\_ostream](#)<\_CharT, \_Traits> \* \_\_tiestr)
- void [unsetf](#) (fmtflags \_\_mask)
- char\_type [widen](#) (char \_\_c) const
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)

- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\_\_pf)([\\_\\_ostream\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_ostream\\_type](#) & [operator<<](#) (ios\_base &(\_\_pf)(ios\_base &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)



### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
  - `void _M_write (const char_type *__s, streamsize __n)`
  - `__ostream_type & write (const char_type *__s, streamsize __n)`
- 
- `operator bool () const`
  - `bool operator! () const`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Static Public Attributes

- `static const fmtflags adjustfield`
- `static const openmode app`
- `static const openmode ate`
- `static const iostate badbit`
- `static const fmtflags basefield`
- `static const seekdir beg`
- `static const openmode binary`
- `static const fmtflags boolalpha`
- `static const seekdir cur`
- `static const fmtflags dec`
- `static const seekdir end`
- `static const iostate eofbit`
- `static const iostate failbit`
- `static const fmtflags fixed`
- `static const fmtflags floatfield`
- `static const iostate goodbit`
- `static const fmtflags hex`
- `static const openmode in`
- `static const fmtflags internal`
- `static const fmtflags left`
- `static const fmtflags oct`
- `static const openmode out`
- `static const fmtflags right`
- `static const fmtflags scientific`
- `static const fmtflags showbase`
- `static const fmtflags showpoint`
- `static const fmtflags showpos`
- `static const fmtflags skipws`
- `static const openmode trunc`
- `static const fmtflags unitbuf`
- `static const fmtflags uppercase`

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename [\\_ValueType](#) >  
  [\\_ostream\\_type](#) & **\_M\_insert** ([\\_ValueType](#) \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) & \_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void **move** ([basic\\_ios](#) & \_\_rhs)
- void **move** ([basic\\_ios](#) && \_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) & \_\_rhs)
- void **swap** ([basic\\_ios](#) & \_\_rhs) noexcept

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 5.634.1 Detailed Description

```
template<typename \_CharT, typename \_Traits, typename \_Alloc>
class std::basic_ostringstream< \_CharT, \_Traits, \_Alloc >
```

Controlling output for std::string.

### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 104 of file `iosfwd`.

## 5.634.2 Member Typedef Documentation

### 5.634.2.1 `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

Definition at line 91 of file `basic_ios.h`.

### 5.634.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 506 of file `ios_base.h`.

### 5.634.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 323 of file `ios_base.h`.

### 5.634.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 398 of file `ios_base.h`.

#### 5.634.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

#### 5.634.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

### 5.634.3 Member Enumeration Documentation

#### 5.634.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.634.4 Constructor &amp; Destructor Documentation

## 5.634.4.1 basic\_ostringstream() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream (
    ios_base::openmode __mode = ios_base::out ) [inline], [explicit]
```

Default constructor starts with an empty string buffer.

## Parameters

<code>__mode</code>	Whether the buffer can read, or write, or both.
---------------------	---

`ios_base::out` is automatically included in *mode*.

Initializes *sb* using *mode*|*out*, and passes *&sb* to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 561 of file *sstream*.

## 5.634.4.2 basic\_ostringstream() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream (
    const __string_type & __str,
    ios_base::openmode __mode = ios_base::out ) [inline], [explicit]
```

Starts with an existing string buffer.

## Parameters

<code>__str</code>	A string to copy as a starting buffer.
<code>__mode</code>	Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes *sb* using *str* and *mode*|*out*, and passes *&sb* to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 579 of file *sstream*.

#### 5.634.4.3 ~basic\_ostringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::~~basic_ostringstream ( ) [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 590 of file sstream.

### 5.634.5 Member Function Documentation

#### 5.634.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

#### 5.634.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

<code>_↵ _s</code>	The array to insert.
<code>_↵ _n</code>	Maximum number of characters to insert.

Definition at line 311 of file ostream.

**5.634.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**5.634.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().



#### 5.634.5.5 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of \_\_rhs into this.

##### Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

##### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

#### 5.634.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

#### 5.634.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 5.634.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 5.634.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::fail ( ) const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::operator bool()`, `std::basic_ios<char, _Traits>::operator!()`, and `std::regex_traits<_CharType>::value()`.

**5.634.5.10** `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**5.634.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

<code>__ch</code>	The new character.
-------------------	--------------------

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**5.634.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**5.634.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file ios\_base.h.

**5.634.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

#### 5.634.5.15 `getloc()`

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, and `std::ws()`.

#### 5.634.5.16 `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

#### 5.634.5.17 `imbue()`

```
template<typename _CharT, typename _Traits>
locale std::basic_ios<_CharT, _Traits>::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

##### Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**5.634.5.18 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
    basic_streambuf<_CharT, _Traits> * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<char, _Traits>::basic_ios()`.

**5.634.5.19 iword()**

```
long& std::ios_base::iword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file ios\_base.h.

#### 5.634.5.20 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>↔

Definition at line 430 of file basic\_ios.h.

#### 5.634.5.21 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file basic\_ios.h.

## 5.634.5.22 operator!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

## 5.634.5.23 operator&lt;&lt;() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(*) (__ostream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

## 5.634.5.24 operator&lt;&lt;() [2/17]

```
template<typename _CharT, typename _Traits>
__ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

## 5.634.5.25 operator&lt;&lt;() [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.


## 5.634.5.26 operator&lt;&lt;() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n ) [inline], [inherited]
```

Integer arithmetic inserters.



**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

**5.634.5.27 operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**5.634.5.28 operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    bool __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

## 5.634.5.29 operator&lt;&lt;() [7/17]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n ) [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

## 5.634.5.30 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

$\leftrightarrow$ <code>__n</code>	A variable of builtin integral type.
---------------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**5.634.5.31 `operator<<()` [9/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    int __n )    [inherited]
```

Integer arithmetic inserters.

**Parameters**

$\leftrightarrow$ <code>__n</code>	A variable of builtin integral type.
---------------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

**5.634.5.32 `operator<<()` [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned int __n )    [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

## 5.634.5.33 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 5.634.5.34 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

$\leftarrow$ $\_n$	A variable of builtin integral type.
-----------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**5.634.5.35 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

$\leftarrow$ $\_$ $\leftarrow$ $\_$ $f$	A variable of builtin floating point type.
---	--

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

**5.634.5.36 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    float __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 5.634.5.37 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 5.634.5.38 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p ) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

<code>_↵ _p</code>	A variable of pointer type.
------------------------	-----------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**5.634.5.39 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

**Parameters**

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

## 5.634.5.40 precision() [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), and std::operator<<().

## 5.634.5.41 precision() [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of precision().

Definition at line 700 of file ios\_base.h.

## 5.634.5.42 put()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c ) [inherited]
```

Simple insertion.

**Parameters**

<code>__c</code>	The character to insert.
------------------	--------------------------



**Returns**

\*this

Tries to insert \_\_c.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

**5.634.5.43 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file ios\_base.h.

**5.634.5.44 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

## Parameters

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

5.634.5.45 `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_ostringstream<_CharT, _Traits, _Alloc>::rdbuf ( ) const [inline]
```

Accessing the underlying buffer.

## Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 630 of file `sstream`.

5.634.5.46 `rdstate()`

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<char>::unset()`.

5.634.5.47 `register_callback()`

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.634.5.48 seekp()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos ) [inherited]
```

Changing the current write position.

**Parameters**

<code>__pos</code>	A file position object.
--------------------	-------------------------

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

**5.634.5.49 seekp()** [2/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current write position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

#### 5.634.5.50 `setf()` [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

##### Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

#### 5.634.5.51 `setf()` [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

##### Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>fmtfl</code> .

##### Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file ios\_base.h.

#### 5.634.5.52 setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

##### Parameters

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

#### 5.634.5.53 str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc >::str ( ) const [inline]
```

Copying out the string buffer.

##### Returns

```
rdbuf()->str()
```

Definition at line 638 of file sstream.

Referenced by `std::__detail::operator<<()`.

#### 5.634.5.54 str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (
    const __string_type & __s ) [inline]
```

Setting a new buffer.

## Parameters

<code>_↵</code>	The string to use as a new sequence.
<code>_s</code>	

Calls `rdbuf() -> str(s)`.

Definition at line 648 of file `sstream`.

## 5.634.5.55 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

## 5.634.5.56 tellp()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ( )
[inherited]
```

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**5.634.5.57** `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**5.634.5.58** `tie()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
    basic_ostream< _CharT, _Traits > * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**5.634.5.59** `unsetf()`

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

## Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

## 5.634.5.60 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

## Parameters

<code>__c</code>	The character to widen.
------------------	-------------------------

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

## 5.634.5.61 width() [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.



**5.634.5.62 width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

**5.634.5.63 write()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n ) [inherited]
```

Character string insertion.

**Parameters**

<code>__s</code>	The array to insert.
<code>__n</code>	Maximum number of characters to insert.

**Returns**

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

#### 5.634.5.64 xalloc()

```
static int std::ios_base::xalloc ( ) throw ( ) [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.634.6 Member Data Documentation

#### 5.634.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.634.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

#### 5.634.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

#### 5.634.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 5.634.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 381 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.634.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.634.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.634.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.634.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.634.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file `ios_base.h`.

#### 5.634.6.11 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.634.6.12 eofbit

```
const iosstate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::eof()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_istream<char>::unget()`, and `std::ws()`.

#### 5.634.6.13 failbit

```
const iosstate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::fail()`, `std::time_get<_CharT, _InIter>::get()`, and `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

#### 5.634.6.14 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file `ios_base.h`.

#### 5.634.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

#### 5.634.6.16 goodbit

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

#### 5.634.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::basic\_ostream< char >::operator<<().

#### 5.634.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for ifstream and fstream.

Definition at line 443 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

#### 5.634.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 340 of file ios\_base.h.

**5.634.6.20 left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

**5.634.6.21 oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::operator<<().

**5.634.6.22 out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for ofstream and fstream.

Definition at line 446 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos().

**5.634.6.23 right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.634.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.634.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

#### 5.634.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.634.6.27 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

#### 5.634.6.28 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.



#### 5.634.6.29 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 449 of file `ios_base.h`.

#### 5.634.6.30 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file `ios_base.h`.

#### 5.634.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

### 5.635 std::basic\_regex<\_Ch\_type, \_Rx\_traits> Class Template Reference

#### Public Types

- typedef [regex\\_constants::syntax\\_option\\_type](#) **flag\_type**
- typedef `traits_type::locale_type` **locale\_type**
- typedef `traits_type::string_type` **string\_type**
- typedef `_Rx_traits` **traits\_type**
- typedef `_Ch_type` **value\_type**

## Public Member Functions

- [basic\\_regex](#) ()
- [basic\\_regex](#) (const \_Ch\_type \* \_\_p, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const \_Ch\_type \* \_\_p, std::size\_t \_\_len, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const [basic\\_regex](#) & \_\_rhs)=default
- [basic\\_regex](#) ([basic\\_regex](#) && \_\_rhs) noexcept=default
- template<typename \_Ch\_traits, typename \_Ch\_alloc >  
  [basic\\_regex](#) (const [std::basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > & \_\_s, [flag\\_type](#) \_\_f=ECMAScript)
- template<typename \_FwdIter >  
  [basic\\_regex](#) (\_FwdIter \_\_first, \_FwdIter \_\_last, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l, [flag\\_type](#) \_\_f=ECMAScript)
- ~[basic\\_regex](#) ()
- [basic\\_regex](#) & [assign](#) (const [basic\\_regex](#) & \_\_rhs)
- [basic\\_regex](#) & [assign](#) ([basic\\_regex](#) && \_\_rhs) noexcept
- [basic\\_regex](#) & [assign](#) (const \_Ch\_type \* \_\_p, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) (const \_Ch\_type \* \_\_p, std::size\_t \_\_len, [flag\\_type](#) \_\_flags)
- template<typename \_Ch\_traits, typename \_Alloc >  
  [basic\\_regex](#) & [assign](#) (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Alloc > & \_\_s, [flag\\_type](#) \_\_flags=ECMAScript)
- template<typename \_InputIterator >  
  [basic\\_regex](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l, [flag\\_type](#) \_\_flags=ECMAScript)
- [flag\\_type](#) [flags](#) () const
- [locale\\_type](#) [getloc](#) () const
- [locale\\_type](#) [imbue](#) ([locale\\_type](#) \_\_loc)
- unsigned int [mark\\_count](#) () const
- [basic\\_regex](#) & [operator=](#) (const [basic\\_regex](#) & \_\_rhs)
- [basic\\_regex](#) & [operator=](#) ([basic\\_regex](#) && \_\_rhs) noexcept
- [basic\\_regex](#) & [operator=](#) (const \_Ch\_type \* \_\_p)
- [basic\\_regex](#) & [operator=](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l)
- template<typename \_Ch\_traits, typename \_Alloc >  
  [basic\\_regex](#) & [operator=](#) (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Alloc > & \_\_s)
- void [swap](#) ([basic\\_regex](#) & \_\_rhs)

## Static Public Attributes

## Constants

*std* [28.8.1](1)

- static constexpr [flag\\_type](#) [icase](#)
- static constexpr [flag\\_type](#) [nosubs](#)
- static constexpr [flag\\_type](#) [optimize](#)
- static constexpr [flag\\_type](#) [collate](#)
- static constexpr [flag\\_type](#) [ECMAScript](#)
- static constexpr [flag\\_type](#) [basic](#)
- static constexpr [flag\\_type](#) [extended](#)
- static constexpr [flag\\_type](#) [awk](#)
- static constexpr [flag\\_type](#) [grep](#)
- static constexpr [flag\\_type](#) [egrep](#)

## Friends

- `template<typename _Bp, typename _Ap, typename _Cp, typename _Rp, __detail::__RegexExecutorPolicy, bool >`  
`bool __detail::__regex_algo_impl ( _Bp, _Bp, match_results< _Bp, _Ap > &, const basic_regex< _Cp, _Rp >`  
`&, regex_constants::match_flag_type)`
- `template<typename, typename, typename, bool >`  
`class __detail::__Executor`

### 5.635.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex< _Ch_type, _Rx_traits >
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 36 of file `regex.h`.

### 5.635.2 Constructor & Destructor Documentation

#### 5.635.2.1 `basic_regex()` [1/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex ( ) [inline]
```

Constructs a basic regular expression that does not match any character sequence.

Definition at line 422 of file `regex.h`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

#### 5.635.2.2 `basic_regex()` [2/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    const _Ch_type * __p,
    flag_type __f = ECMAScript ) [inline], [explicit]
```

Constructs a basic regular expression from the sequence `[__p, __p + char_traits<_Ch_type>::length(__p))` interpreted according to the flags in `__f`.

## Parameters

<code>__p</code>	A pointer to the start of a C-style null-terminated string containing a regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

## Exceptions

<code>regex_error</code>	if <code>__p</code> is not a valid regular expression.
--------------------------	--

Definition at line 438 of file regex.h.

## 5.635.2.3 basic\_regex() [3/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    const _Ch_type * __p,
    std::size_t __len,
    flag_type __f = ECMAScript ) [inline]
```

Constructs a basic regular expression from the sequence `[p, p + len)` interpreted according to the flags in `f`.

## Parameters

<code>__p</code>	A pointer to the start of a string containing a regular expression.
<code>__len</code>	The length of the string containing the regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

## Exceptions

<code>regex_error</code>	if <code>__p</code> is not a valid regular expression.
--------------------------	--

Definition at line 454 of file regex.h.

## 5.635.2.4 basic\_regex() [4/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    const basic_regex< _Ch_type, _Rx_traits > & __rhs ) [default]
```

Copy-constructs a basic regular expression.

## Parameters

<code>__rhs</code>	A regex object.
--------------------	-----------------

5.635.2.5 `basic_regex()` [5/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    basic_regex< _Ch_type, _Rx_traits > && __rhs ) [default], [noexcept]
```

Move-constructs a basic regular expression.

## Parameters

<code>__rhs</code>	A regex object.
--------------------	-----------------

5.635.2.6 `basic_regex()` [6/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits , typename _Ch_alloc >
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    flag_type __f = ECMAScript ) [inline], [explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

## Parameters

<code>__s</code>	A string containing a regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

## Exceptions

<code>regex_error</code>	if <code>__s</code> is not a valid regular expression.
--------------------------	--

Definition at line 484 of file `regex.h`.

## 5.635.2.7 basic\_regex() [7/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _FwdIter >
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    _FwdIter __first,
    _FwdIter __last,
    flag_type __f = ECMAScript ) [inline]
```

Constructs a basic regular expression from the range [first, last) interpreted according to the flags in f.

## Parameters

<code>__first</code>	The start of a range containing a valid regular expression.
<code>__last</code>	The end of a range containing a valid regular expression.
<code>__f</code>	The format flags of the regular expression.

## Exceptions

<code>regex_error</code>	if [ <code>__first</code> , <code>__last</code> ) is not a valid regular expression.
--------------------------	--

Definition at line 504 of file regex.h.

## 5.635.2.8 basic\_regex() [8/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
    initializer_list<_Ch_type > __l,
    flag_type __f = ECMAScript ) [inline]
```

Constructs a basic regular expression from an initializer list.

## Parameters

<code>__l</code>	The initializer list.
<code>__f</code>	The format flags of the regular expression.

## Exceptions

<code>regex_error</code>	if <code>__l</code> is not a valid regular expression.
--------------------------	--

Definition at line 517 of file `regex.h`.

### 5.635.2.9 `~basic_regex()`

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::~~basic_regex ( ) [inline]
```

Destroys a basic regular expression.

Definition at line 524 of file `regex.h`.

## 5.635.3 Member Function Documentation

### 5.635.3.1 `assign()` [1/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
    const basic_regex< _Ch_type, _Rx_traits > & __rhs ) [inline]
```

the real assignment operator.

#### Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 582 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::operator=()`.

### 5.635.3.2 `assign()` [2/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
    basic_regex< _Ch_type, _Rx_traits > && __rhs ) [inline], [noexcept]
```

The move-assignment operator.

## Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 595 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

## 5.635.3.3 assign() [3/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
    const _Ch_type * __p,
    flag_type __flags = ECMAScript ) [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

## Parameters

<code>__p</code>	A pointer to a C-style null-terminated string containing a regular expression pattern.
<code>__flags</code>	Syntax option flags.

## Exceptions

<code>regex_error</code>	if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 616 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

## 5.635.3.4 assign() [4/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
    const _Ch_type * __p,
    std::size_t __len,
    flag_type __flags ) [inline]
```

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.



## Parameters

<code>__p</code>	A pointer to a C-style string containing a regular expression pattern.
<code>__len</code>	The length of the regular expression pattern string.
<code>__flags</code>	Syntax option flags.

## Exceptions

<code>regex_error</code>	if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 633 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

5.635.3.5 `assign()` [5/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc >
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
    const basic_string<_Ch_type, _Ch_traits, _Alloc > & __s,
    flag_type __flags = ECMAScript ) [inline]
```

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

## Parameters

<code>__s</code>	A string containing a regular expression pattern.
<code>__flags</code>	Syntax option flags.

## Exceptions

<code>regex_error</code>	if <code>__s</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 649 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`, `std::basic_regex<_Ch_type, _Rx_traits >::basic_regex()`, `std::basic_string<_CharT, _Traits, _Alloc >::data()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

5.635.3.6 `assign()` [6/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _InputIterator >
```

```
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
    _InputIterator __first,
    _InputIterator __last,
    flag_type __flags = ECMAScript ) [inline]
```

Assigns a new regular expression to a regex object.

#### Parameters

<code>__first</code>	The start of a range containing a valid regular expression.
<code>__last</code>	The end of a range containing a valid regular expression.
<code>__flags</code>	Syntax option flags.

#### Exceptions

<code>regex_error</code>	if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged.
--------------------------	--

Definition at line 671 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

#### 5.635.3.7 assign() [7/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
    initializer_list<_Ch_type > __l,
    flag_type __flags = ECMAScript ) [inline]
```

Assigns a new regular expression to a regex object.

#### Parameters

<code>__l</code>	An initializer list representing a regular expression.
<code>__flags</code>	Syntax option flags.

#### Exceptions

<code>regex_error</code>	if <code>__l</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged.
--------------------------	--

Definition at line 687 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

#### 5.635.3.8 flags()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
flag_type std::basic_regex<_Ch_type, _Rx_traits >::flags ( ) const [inline]
```

Gets the flags used to construct the regular expression or in the last call to assign().

Definition at line 708 of file regex.h.

#### 5.635.3.9 getloc()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits >::getloc ( ) const [inline]
```

Gets the locale currently imbued in the regular expression object.

Definition at line 730 of file regex.h.

#### 5.635.3.10 imbue()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits >::imbue (
    locale_type __loc ) [inline]
```

Imbues the regular expression object with the given locale.

##### Parameters

<code>__loc</code>	A locale.
--------------------	-----------

Definition at line 718 of file regex.h.

#### 5.635.3.11 mark\_count()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
unsigned int std::basic_regex<_Ch_type, _Rx_traits >::mark_count ( ) const [inline]
```

Gets the number of marked subexpressions within the regular expression.

Definition at line 696 of file regex.h.

**5.635.3.12 operator=()** [1/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    const basic_regex<_Ch_type, _Rx_traits > & __rhs ) [inline]
```

Assigns one regular expression to another.

Definition at line 531 of file regex.h.

References std::basic\_regex<\_Ch\_type, \_Rx\_traits >::assign().

**5.635.3.13 operator=()** [2/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    basic_regex<_Ch_type, _Rx_traits > && __rhs ) [inline], [noexcept]
```

Move-assigns one regular expression to another.

Definition at line 538 of file regex.h.


References std::basic\_regex<\_Ch\_type, \_Rx\_traits >::assign().

**5.635.3.14 operator=()** [3/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    const _Ch_type * __p ) [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

**Parameters**

 <b>__p</b>	A pointer to the start of a null-terminated C-style string containing a regular expression.
--	---

Definition at line 549 of file regex.h.

References std::basic\_regex<\_Ch\_type, \_Rx\_traits >::assign().

**5.635.3.15 operator=()** [4/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    initializer_list<_Ch_type > __l ) [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

#### Parameters

↩	The initializer list.
↩	
↩	
↩	
↩	

#### Exceptions

<i>regex_error</i>	if <code>__l</code> is not a valid regular expression.
--------------------	--

Definition at line 561 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

#### 5.635.3.16 `operator=()` [5/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc >
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::operator= (
    const basic_string<_Ch_type, _Ch_traits, _Alloc > & __s ) [inline]
```

Replaces a regular expression with a new one constructed from a string.

#### Parameters

↩	A pointer to a string containing a regular expression.
<code>__s</code>	

Definition at line 572 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

#### 5.635.3.17 `swap()`

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
void std::basic_regex<_Ch_type, _Rx_traits >::swap (
    basic_regex<_Ch_type, _Rx_traits > & __rhs ) [inline]
```

Swaps the contents of two regular expression objects.

## Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 740 of file regex.h.

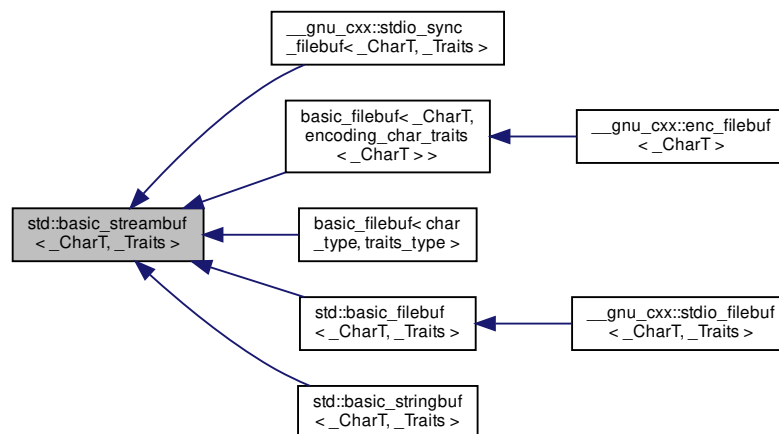
Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 5.636 std::basic\_streambuf&lt;\_CharT, \_Traits&gt; Class Template Reference

Inheritance diagram for `std::basic_streambuf<_CharT, _Traits>`:



## Public Types

- typedef `_CharT` [char\\_type](#)
  - typedef `_Traits` [traits\\_type](#)
  - typedef `traits_type::int_type` [int\\_type](#)
  - typedef `traits_type::pos_type` [pos\\_type](#)
  - typedef `traits_type::off_type` [off\\_type](#)
- 
- typedef `basic_streambuf<char_type, traits_type>` [\\_\\_streambuf\\_type](#)

## Public Member Functions

- virtual `~basic_streambuf()`
  - `locale getloc()` const
  - `streamsize in_avail()`
  - `locale pubimbue(const locale &__loc)`
  - `int_type sbumpc()`
  - `int_type sgetc()`
  - `streamsize sgetn(char_type *__s, streamsize __n)`
  - `int_type snextc()`
  - `int_type sputbackc(char_type __c)`
  - `int_type sputc(char_type __c)`
  - `streamsize sputn(const char_type *__s, streamsize __n)`
  - `int_type sungetc()`
- 
- `basic_streambuf * pubsetbuf(char_type *__s, streamsize __n)`
  - `pos_type pubseekoff(off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `pos_type pubseekpos(pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `int pubsync()`

## Protected Member Functions

- `basic_streambuf()`
- `basic_streambuf(const basic_streambuf &)`
- `void __safe_gbump(streamsize __n)`
- `void __safe_pbump(streamsize __n)`
- `void gbump(int __n)`
- virtual `void imbue(const locale &__loc __attribute__((__unused__)))`
- `basic_streambuf & operator= (const basic_streambuf &)`
- virtual `int_type overflow(int_type __c __attribute__((__unused__))=traits_type::eof())`
- virtual `int_type pbackfail(int_type __c __attribute__((__unused__))=traits_type::eof())`
- `void pbump(int __n)`
- virtual `pos_type seekoff(off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)`
- virtual `pos_type seekpos(pos_type, ios_base::openmode=ios_base::in|ios_base::out)`
- virtual `basic_streambuf< char_type, _Traits > * setbuf(char_type *, streamsize)`
- `void setg(char_type * __gbeg, char_type * __gnext, char_type * __gend)`
- `void setp(char_type * __pbeg, char_type * __pend)`
- virtual `streamsize showmanyc()`
- `void swap(basic_streambuf &__sb)`
- virtual `int sync()`
- virtual `int_type uflow()`
- virtual `int_type underflow()`
- virtual `streamsize xsgetn(char_type *__s, streamsize __n)`
- virtual `streamsize xsputn(const char_type *__s, streamsize __n)`

- [char\\_type \\* eback](#) () const
- [char\\_type \\* gptr](#) () const
- [char\\_type \\* egptr](#) () const

- [char\\_type \\* pbase](#) () const
- [char\\_type \\* pptr](#) () const
- [char\\_type \\* ep\\_ptr](#) () const

#### Protected Attributes

- [locale \\_M\\_buf\\_locale](#)
- [char\\_type \\* \\_M\\_in\\_beg](#)
- [char\\_type \\* \\_M\\_in\\_cur](#)
- [char\\_type \\* \\_M\\_in\\_end](#)
- [char\\_type \\* \\_M\\_out\\_beg](#)
- [char\\_type \\* \\_M\\_out\\_cur](#)
- [char\\_type \\* \\_M\\_out\\_end](#)

#### Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof(basic_streambuf *, basic_streambuf *, bool &)`
- `template<typename _CharT2, typename _Distance >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type advance(istreambuf_iterator< _CharT2 > &, _Distance)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`
- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type find`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > &getline(basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2,`  
`_Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, _CharT2 *)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, basic_string<`  
`_CharT2, _Traits2, _Alloc > &)`
- `class ostreambuf_iterator< char_type, traits_type >`

#### 5.636.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_streambuf< _CharT, _Traits >
```

The actual work of input and output (interface).



## Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

- Stream buffers can impose various constraints on the sequences they control. Some constraints are:
  - The controlled input sequence can be not readable.
  - The controlled output sequence can be not writable.
  - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
  - The controlled sequences can support operations *directly* to or from associated sequences.
  - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
- Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
  - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
- The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
  - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and  $xbeg < xnext$  for an input sequence, then a *putback position* is available. In this case, *xnext[-1]* shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 80 of file `iosfwd`.

### 5.636.2 Member Typedef Documentation

#### 5.636.2.1 \_\_streambuf\_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits>::__streambuf_type
```

This is a non-standard type.

Definition at line 140 of file streambuf.

#### 5.636.2.2 char\_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 131 of file streambuf.

#### 5.636.2.3 int\_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf<_CharT, _Traits>::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 133 of file streambuf.

#### 5.636.2.4 off\_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf<_CharT, _Traits>::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 135 of file streambuf.

### 5.636.2.5 pos\_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 134 of file streambuf.

### 5.636.2.6 traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 132 of file streambuf.

## 5.636.3 Constructor & Destructor Documentation

### 5.636.3.1 ~basic\_streambuf()

```
template<typename _CharT, typename _Traits>
virtual std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf ( ) [inline], [virtual]
```

Destructor deallocates no buffer space.

Definition at line 204 of file streambuf.

### 5.636.3.2 basic\_streambuf()

```
template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf ( ) [inline], [protected]
```

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the basic\_streambuf class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 470 of file streambuf.

#### 5.636.4 Member Function Documentation

##### 5.636.4.1 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 489 of file streambuf.

##### 5.636.4.2 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

#### 5.636.4.3 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 5.636.4.4 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
    int __n ) [inline], [protected]
```

Moving the read position.

##### Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

#### 5.636.4.5 getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline]
```

Locale access.

##### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

## 5.636.4.6 gptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

## 5.636.4.7 imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale &__loc __attribute__((unused)) ) [inline], [protected], [virtual]
```

Changes translations.

## Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Definition at line 583 of file streambuf.

## 5.636.4.8 in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

**5.636.4.9 overflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 775 of file `streambuf`.

**5.636.4.10 pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual]
```

Tries to back up the input sequence.

## Parameters

<code>_c</code>	The character to be inserted back into the sequence.
-----------------	--

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Definition at line 731 of file `streambuf`.

## 5.636.4.11 pbase()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase ( ) const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

## 5.636.4.12 pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
    int __n ) [inline], [protected]
```

Moving the write position.



**Parameters**

<code>_↵</code>	The delta by which to move.
<code>_n</code>	

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**5.636.4.13 pptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**5.636.4.14 pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
    const locale & __loc ) [inline]
```

Entry point for imbue().

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

## 5.636.4.15 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline]
```

Alters the stream position.

## Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

## 5.636.4.16 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline]
```

Alters the stream position.

## Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

## 5.636.4.17 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 5.636.4.18 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 5.636.4.19 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline]
```

Getting the next character.

##### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 5.636.4.20 `seekoff()`

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 609 of file `streambuf`.

## 5.636.4.21 seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

## Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 621 of file `streambuf`.

## 5.636.4.22 setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf<char_type,_Traits>* std::basic_streambuf< _CharT, _Traits >::setbuf (
    char_type * ,
    streamsize ) [inline], [protected], [virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

## Note

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 598 of file `streambuf`.

## 5.636.4.23 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected]
```

Setting the three read area pointers.

**Parameters**

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file `streambuf`.

**5.636.4.24 setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend ) [inline], [protected]
```

Setting the three write area pointers.

**Parameters**

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file `streambuf`.

**5.636.4.25 sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

## 5.636.4.26 sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits >::sgetn (
    char_type * __s,
    streamsize __n ) [inline]
```

Entry point for xsgetn.

## Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

## 5.636.4.27 showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf<_CharT, _Traits >::showmanyc ( ) [inline], [protected],
[virtual]
```

Investigating the data available.

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits >`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT > >`, `std::basic_filebuf<char_type, traits_type >`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc >`.

Definition at line 656 of file `streambuf`.

**5.636.4.28** `snextc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**5.636.4.29** `sputbackc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
    char_type __c ) [inline]
```

Pushing characters back into the input stream.

**Parameters**

<code>__c</code>	The character to push back.
------------------	-----------------------------

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**5.636.4.30** `sputc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
    char_type __c ) [inline]
```

Entry point for all single-character output functions.

## Parameters

<code>__c</code>	A character to output.
------------------	------------------------

## Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

## 5.636.4.31 sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
    const char_type * __s,
    streamsize __n ) [inline]
```

Entry point for all single-character output functions.

## Parameters

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

## 5.636.4.32 sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline]
```

Moving backwards in the input stream.



**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**5.636.4.33 sync()**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync (
    void ) [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, and `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 634 of file `streambuf`.

**5.636.4.34 uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline], [protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

## 5.636.4.35 underflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::underflow ( ) [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

## Note

Base class version does nothing, returns eof().

Reimplemented in `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_type, traits_type>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, and `__gnu_cxx::stdio_sync_filebuf<_CharT, traits_type, _Alloc>`.

Definition at line 694 of file streambuf.

## 5.636.4.36 xsgetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_streambuf<_CharT, _Traits>::xsgetn (
    char_type * __s,
    streamsize __n ) [protected], [virtual]
```

Multiple character extraction.

## Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 46 of file `streambuf.tcc`.

**5.636.4.37 xspn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xspn (
    const char_type * __s,
    streamsize __n) [protected], [virtual]
```

Multiple character insertion.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 80 of file `streambuf.tcc`.

**5.636.5 Member Data Documentation**

#### 5.636.5.1 \_M\_buf\_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected]
```

Current locale setting.

Definition at line 199 of file streambuf.

#### 5.636.5.2 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 5.636.5.3 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected]
```

Current read area.

Definition at line 192 of file streambuf.

#### 5.636.5.4 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected]
```

End of get area.

Definition at line 193 of file streambuf.

#### 5.636.5.5 \_M\_out\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 5.636.5.6 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected]
```

Current put area.

Definition at line 195 of file streambuf.

#### 5.636.5.7 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected]
```

End of put area.

Definition at line 196 of file streambuf.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [streambuf](#)
- [streambuf.tcc](#)

### 5.637 `std::basic_string< _CharT, _Traits, _Alloc >` Class Template Reference

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic\_string >` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse\_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic\_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse\_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- [basic\\_string](#) ()
- [basic\\_string](#) (const \_Alloc &\_\_a)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- [basic\\_string](#) (const \_CharT \*\_\_s, size\_type \_\_n, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const \_CharT \*\_\_s, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) ([basic\\_string](#) &&\_\_str) noexcept
- [basic\\_string](#) (initializer\_list<\_CharT> \_\_l, const \_Alloc &\_\_a=\_Alloc())
- template<class \_InputIterator>
 [basic\\_string](#) (\_InputIterator \_\_beg, \_InputIterator \_\_end, const \_Alloc &\_\_a=\_Alloc())
- [~basic\\_string](#) () noexcept
- template<typename \_InputIterator>
 [basic\\_string](#)<\_CharT, \_Traits, \_Alloc> & **[M\\_replace\\_dispatch](#)** (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2, \_\_false\_type)
- template<typename \_InIterator>
 \_CharT \* **[S\\_construct](#)** (\_InIterator \_\_beg, \_InIterator \_\_end, const \_Alloc &\_\_a, [forward\\_iterator\\_tag](#))
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [append](#) (size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [append](#) (initializer\_list<\_CharT> \_\_l)
- template<class \_InputIterator>
 [basic\\_string](#) & [append](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [assign](#) (size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator>
 [basic\\_string](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & [assign](#) (initializer\_list<\_CharT> \_\_l)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [at](#) (size\_type \_\_n)
- reference [back](#) ()
- const\_reference [back](#) () const noexcept
- iterator [begin](#) ()
- const\_iterator [begin](#) () const noexcept
- const \_CharT \* [c\\_str](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- int [compare](#) (const [basic\\_string](#) &\_\_str) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str) const

- int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- int `compare` (const \_CharT \*\_\_s) const noexcept
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- size\_type `copy` (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- `const_reverse_iterator` `crbegin` () const noexcept
- `const_reverse_iterator` `crend` () const noexcept
- const \_CharT \* `data` () const noexcept
- bool `empty` () const noexcept
- iterator `end` ()
- const\_iterator `end` () const noexcept
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=`npos`)
- iterator `erase` (iterator \_\_position)
- iterator `erase` (iterator \_\_first, iterator \_\_last)
- size\_type `find` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const noexcept
- size\_type `find` (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_first_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_first_of` (const \_CharT \*\_\_s, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_last_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=`npos`) const noexcept
- size\_type `find_last_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_last_not_of` (const \_CharT \*\_\_s, size\_type \_\_pos=`npos`) const noexcept
- size\_type `find_last_not_of` (\_CharT \_\_c, size\_type \_\_pos=`npos`) const noexcept
- size\_type `find_last_of` (const `basic_string` &\_\_str, size\_type \_\_pos=`npos`) const noexcept
- size\_type `find_last_of` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_last_of` (const \_CharT \*\_\_s, size\_type \_\_pos=`npos`) const noexcept
- size\_type `find_last_of` (\_CharT \_\_c, size\_type \_\_pos=`npos`) const noexcept
- reference `front` ()
- const\_reference `front` () const noexcept
- allocator\_type `get_allocator` () const noexcept
- void `insert` (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator >  
void `insert` (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void `insert` (iterator \_\_p, `initializer_list`< \_CharT > \_\_l)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` &\_\_str)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n=`npos`)
- `basic_string` & `insert` (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string` & `insert` (size\_type \_\_pos, const \_CharT \*\_\_s)
- `basic_string` & `insert` (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- iterator `insert` (iterator \_\_p, \_CharT \_\_c)
- size\_type `length` () const noexcept
- size\_type `max_size` () const noexcept
- `basic_string` & `operator+=` (const `basic_string` &\_\_str)

- [basic\\_string](#) & [operator+=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) (initializer\_list<\_CharT> \_\_l)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) && \_\_str)
- [basic\\_string](#) & [operator=](#) (initializer\_list<\_CharT> \_\_l)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept
- reference [operator\[\]](#) (size\_type \_\_pos)
- void [pop\\_back](#) ()
- void [push\\_back](#) (\_CharT \_\_c)
- [reverse\\_iterator](#) [rbegin](#) ()
- const\_reverse\_iterator [rbegin](#) () const noexcept
- [reverse\\_iterator](#) [rend](#) ()
- const\_reverse\_iterator [rend](#) () const noexcept
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n2=[npos](#))
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator>  
[basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, initializer\_list<\_CharT> \_\_l)
- void [reserve](#) (size\_type \_\_res\_arg=0)
- void [resize](#) (size\_type \_\_n, \_CharT \_\_c)
- void [resize](#) (size\_type \_\_n)
- size\_type [rfind](#) (const [basic\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- void [shrink\\_to\\_fit](#) () noexcept
- size\_type [size](#) () const noexcept
- [basic\\_string](#) substr (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- void [swap](#) ([basic\\_string](#) & \_\_s)

#### Static Public Attributes

- static const size\_type [npos](#)



### 5.637.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects.

#### Template Parameters

<code>_CharT</code>	Type of character
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

**Todo** \nNeeds documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[basic_string<char_type>]           _M_capacity
_M_dataplus                        _M_refcount
_M_p ----->                     unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string<_Rep>::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 3108 of file `basic_string.h`.

### 5.637.2 Constructor & Destructor Documentation

#### 5.637.2.1 basic\_string() [1/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ( ) [inline]
```

Default constructor creates an empty string.

Definition at line 3493 of file basic\_string.h.

Referenced by std::basic\_string< char >::substr().

#### 5.637.2.2 basic\_string() [2/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _Alloc & __a ) [explicit]
```

Construct an empty string using allocator *a*.

Definition at line 618 of file basic\_string.tcc.

#### 5.637.2.3 basic\_string() [3/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str )
```

Construct string with copy of value of *str*.

##### Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 610 of file basic\_string.tcc.

#### 5.637.2.4 basic\_string() [4/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    const _Alloc & __a = _Alloc() )
```

Construct string as copy of a substring.

**Parameters**

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__a</code>	Allocator to use.

Definition at line 624 of file `basic_string.tcc`.

**5.637.2.5 `basic_string()`** [5/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n )
```

Construct string as copy of a substring.

**Parameters**

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.

Definition at line 634 of file `basic_string.tcc`.

**5.637.2.6 `basic_string()`** [6/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n,
    const _Alloc & __a )
```

Construct string as copy of a substring.

**Parameters**

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.




Definition at line 644 of file basic\_string.tcc.

#### 5.637.2.7 basic\_string() [7/12]

```
template<typename _CharT, typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _CharT * __s,
    size_type __n,
    const _Alloc & __a = _Alloc() )
```

Construct string initialized by a character array.

##### Parameters

 __s	Source character array.
 __n	Number of characters to copy.
 __a	Allocator to use (default is default allocator).

NB: \_\_s must have at least \_\_n characters, '\0' has no special meaning.



Definition at line 656 of file basic\_string.tcc.

#### 5.637.2.8 basic\_string() [8/12]

```
template<typename _CharT, typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    const _CharT * __s,
    const _Alloc & __a = _Alloc() )
```

Construct string as copy of a C string.

##### Parameters

 __s	Source C string.
 __a	Allocator to use (default is default allocator).

Definition at line 663 of file basic\_string.tcc.

**5.637.2.9 basic\_string()** [9/12]

```
template<typename _CharT, typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    size_type __n,
    _CharT __c,
    const _Alloc & __a = _Alloc() )
```

Construct string as multiple characters.

**Parameters**

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 670 of file basic\_string.tcc.

**5.637.2.10 basic\_string()** [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    basic_string< _CharT, _Traits, _Alloc > && __str ) [inline], [noexcept]
```

Move construct string.

**Parameters**

<code>__str</code>	Source string.
--------------------	----------------

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 3575 of file basic\_string.h.

**5.637.2.11 basic\_string()** [11/12]

```
template<typename _CharT, typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    initializer_list< _CharT > __l,
    const _Alloc & __a = _Alloc() )
```

Construct string from an initializer list.

## Parameters

<code>_↵ _l</code>	std::initializer_list of characters.
<code>_↵ _a</code>	Allocator to use (default is default allocator).

Definition at line 685 of file basic\_string.tcc.

## 5.637.2.12 basic\_string() [12/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
template<typename _InputIterator >
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
    _InputIterator __beg,
    _InputIterator __end,
    const _Alloc & __a = _Alloc() )
```

Construct string as copy of a range.

## Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 678 of file basic\_string.tcc.

## 5.637.2.13 ~basic\_string()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ( ) [inline], [noexcept]
```

Destroy the string instance.

Definition at line 3643 of file basic\_string.h.

## 5.637.3 Member Function Documentation

## 5.637.3.1 append() [1/7]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const basic_string< _CharT, _Traits, _Alloc > & __str )
```

Append a string to this string.

**Parameters**

<code>__str</code>	The string to append.
--------------------	-----------------------

**Returns**

Reference to this string.

Definition at line 775 of file `basic_string.tcc`.

Referenced by `std::basic_string< char >::append()`, and `std::basic_string< char >::operator+=()`.

**5.637.3.2 `append()`** [2/7]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos )
```

Append a substring.

**Parameters**

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 792 of file `basic_string.tcc`.

## 5.637.3.3 append() [3/7]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    const _CharT * __s,
    size_type __n )
```

Append a C substring.

## Parameters

<code>__s</code>	The C string to append.
<code>__n</code>	The number of characters to append.

## Returns

Reference to this string.

Definition at line 748 of file basic\_string.tcc.

## 5.637.3.4 append() [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (
    const _CharT * __s ) [inline]
```

Append a C string.

## Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

## Returns

Reference to this string.

Definition at line 4178 of file basic\_string.h.





### 5.637.3.5 `append()` [5/7]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
    size_type __n,
    _CharT __c )
```

Append multiple characters.

#### Parameters

 <code>__n</code>	The number of characters to append.
 <code>__c</code>	The character to use.

#### Returns

Reference to this string.

Appends `__n` copies of `__c` to this string.





Definition at line 731 of file `basic_string.tcc`.

### 5.637.3.6 `append()` [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (
    initializer_list< _CharT > __l ) [inline]
```

Append an `initializer_list` of characters.

#### Parameters

    <code>l</code>	The <code>initializer_list</code> of characters to append.
--	--

#### Returns

Reference to this string.

Definition at line 4202 of file `basic_string.h`.

### 5.637.3.7 append() [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Append a range of characters.

#### Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

#### Returns

Reference to this string.

Appends characters in the range [`__first`,`__last`) to this string.

Definition at line 4216 of file `basic_string.h`.

### 5.637.3.8 assign() [1/8]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::assign (
    const basic_string<_CharT, _Traits, _Alloc> & __str )
```

Set value to contents of another string.

#### Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

#### Returns

Reference to this string.

Definition at line 693 of file `basic_string.tcc`.

Referenced by `std::basic_string<char>::assign()`, and `std::basic_string<char>::operator=()`.

**5.637.3.9** `assign()` [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
    basic_string< _CharT, _Traits, _Alloc > && __str ) [inline]
```

Set value to contents of another string.

**Parameters**

<code>__str</code>	Source string to use.
--------------------	-----------------------

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 4285 of file `basic_string.h`.

**5.637.3.10** `assign()` [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos,
    size_type __n = npos ) [inline]
```

Set value to a substring of a string.

**Parameters**

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the

number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 4306 of file `basic_string.h`.

#### 5.637.3.11 `assign()` [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::assign (
    const _CharT * __s,
    size_type __n )
```

Set value to a C substring.

##### Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

##### Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 709 of file `basic_string.tcc`.

#### 5.637.3.12 `assign()` [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign (
    const _CharT * __s ) [inline]
```

Set value to contents of a C string.

##### Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 4334 of file `basic_string.h`.

**5.637.3.13 `assign()`** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
    size_type __n,
    _CharT __c ) [inline]
```

Set value to multiple characters.

**Parameters**

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 4350 of file `basic_string.h`.

**5.637.3.14 `assign()`** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Set value to a range of characters.

**Parameters**

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

**Returns**

Reference to this string.

Sets value of string to characters in the range [`__first`,`__last`).

Definition at line 4363 of file `basic_string.h`.

**5.637.3.15 assign()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign (
    initializer_list<_CharT > __l ) [inline]
```

Set value to an `initializer_list` of characters.

**Parameters**

<code>↔</code>	The <code>initializer_list</code> of characters to assign.
<code>↔</code>	
<code>↔</code>	
<code>↔</code>	
<code>/</code>	

**Returns**

Reference to this string.

Definition at line 4373 of file `basic_string.h`.

**5.637.3.16 at()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc >::at (
    size_type __n ) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

<code>↔</code>	The index of the character to access.
<code>__n</code>	

**Returns**

Read-only (const) reference to the character.

**Exceptions**

<code>std::out_of_range</code>	If <i>n</i> is an invalid index.
--------------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 4006 of file `basic_string.h`.

**5.637.3.17 at()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
    size_type __n ) [inline]
```

Provides access to the data contained in the string.

**Parameters**

<code>_↔</code>	The index of the character to access.
<code>_n</code>	

**Returns**

Read/write reference to the character.

**Exceptions**

<code>std::out_of_range</code>	If <i>n</i> is an invalid index.
--------------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 4028 of file `basic_string.h`.

**5.637.3.18 back()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) [inline]
```

Returns a read/write reference to the data at the last element of the string.

Definition at line 4067 of file basic\_string.h.

#### 5.637.3.19 back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 4078 of file basic\_string.h.

#### 5.637.3.20 begin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) [inline]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 3729 of file basic\_string.h.

Referenced by std::basic\_string< char >::crend(), std::regex\_match(), std::regex\_replace(), std::regex\_search(), and std::basic\_string< char >::rend().

#### 5.637.3.21 begin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 3740 of file basic\_string.h.

#### 5.637.3.22 c\_str()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str ( ) const [inline], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 5132 of file basic\_string.h.

Referenced by std::collate< \_CharT >::do\_compare(), std::money\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::collate< \_CharT >::do\_transform(), and std::regex\_replace().



**5.637.3.23 capacity()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity ( ) const [inline], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 3902 of file basic\_string.h.

Referenced by `std::basic_string< char >::push_back()`, and `std::basic_string< char >::shrink_to_fit()`.

**5.637.3.24 cbegin()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 3804 of file basic\_string.h.

**5.637.3.25 cend()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 3812 of file basic\_string.h.

**5.637.3.26 clear()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases the string, making it empty.

Definition at line 3930 of file basic\_string.h.

**5.637.3.27 compare()** [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    const basic_string< _CharT, _Traits, _Alloc > & __str ) const [inline]
```

Compare to a string.

## Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

## Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 5688 of file `basic_string.h`.

Referenced by `std::sub_match<_Bi_iter >::compare()`.

## 5.637.3.28 compare() [2/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Alloc > & __str ) const
```

Compare substring to a string.

## Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1385 of file `basic_string.tcc`.

**5.637.3.29 compare()** [3/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    size_type __pos1,
    size_type __n1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n2 = npos ) const
```

Compare substring to a substring.

**Parameters**

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of str.
<code>__n2</code>	Number of characters in substring of str.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1400 of file `basic_string.tcc`.

**5.637.3.30 compare()** [4/6]

```
template<typename _CharT, typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    const _CharT * __s ) const [noexcept]
```

Compare to a C string.

**Parameters**

<code>__s</code>	C string to compare against.
------------------	------------------------------

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before \_\_s, 0 if their values are equivalent, or > 0 if this string is ordered after \_\_s. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from \_\_s. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1418 of file basic\_string.tcc.

**5.637.3.31 compare()** [5/6]

```
template<typename _CharT, typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s ) const
```

Compare substring to a C string.

**Parameters**

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the \_\_n1 characters starting at pos. Returns an integer < 0 if the substring is ordered before \_\_s, 0 if their values are equivalent, or > 0 if the substring is ordered after \_\_s. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from \_\_s. The function then compares the two string by calling traits::compare(substring.data(),\_\_s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1433 of file basic\_string.tcc.

**5.637.3.32 compare()** [6/6]

```
template<typename _CharT, typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2 ) const
```

Compare substring against a character array.

**Parameters**

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of s.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

Definition at line 1449 of file `basic_string.tcc`.

**5.637.3.33 copy()**

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
    _CharT * __s,
    size_type __n,
    size_type __pos = 0 ) const
```

Copy substring into C string.

**Parameters**

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

**Returns**

Number of characters actually copied

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
--------------------------------	-------------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 1143 of file `basic_string.tcc`.

#### 5.637.3.34 crbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crbegin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 3821 of file `basic_string.h`.

#### 5.637.3.35 crend()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 3830 of file `basic_string.h`.

#### 5.637.3.36 data()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::data ( ) const [inline], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

Definition at line 5144 of file `basic_string.h`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, `std::basic_string<char>::compare()`, `std::collate<_CharT>::do_compare()`, `std::collate<_CharT>::do_transform()`, `std::match_results<_Bi_iter>::format()`, `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc>::from_bytes()`, `std::operator<()`, `std::operator==()`, `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc>::to_bytes()`, and `std::regex_traits<_CharType>::transform()`.

**5.637.3.37 empty()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 3952 of file `basic_string.h`.

**5.637.3.38 end()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ( ) [inline]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 3748 of file `basic_string.h`.

Referenced by `std::basic_string< char >::rbegin()`, `std::basic_string< char >::rbegin()`, `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

**5.637.3.39 end()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 3759 of file `basic_string.h`.

**5.637.3.40 erase()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::erase (
    size_type __pos = 0,
    size_type __n = npos ) [inline]
```

Remove characters.

**Parameters**

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <i>pos</i> is beyond the end of this string.
--------------------------------	---

Removes *\_\_n* characters from this string starting at *\_\_pos*. The length of the string is reduced by *\_\_n*. If there are *< \_\_n* characters to remove, the remainder of the string is truncated. If *\_\_p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4636 of file `basic_string.h`.

**5.637.3.41 erase()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc >::erase (
    iterator __position ) [inline]
```

Remove one character.

**Parameters**

<i>__position</i>	Iterator referencing the character to remove.
-------------------	---

**Returns**

iterator referencing same location after removal.

Removes the character at *\_\_position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 4652 of file `basic_string.h`.

**5.637.3.42 erase()** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::iterator std::basic_string<_CharT, _Traits, _Alloc >←
::erase (
    iterator __first,
    iterator __last )
```

Remove a range of characters.



**Parameters**

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 841 of file basic\_string.tcc.

**5.637.3.43 find()** [1/4]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const [noexcept]
```

Find position of a C substring.

**Parameters**

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1188 of file basic\_string.tcc.

Referenced by `std::basic_string< char >::find()`, and `std::basic_string< char >::find_first_of()`.

## 5.637.3.44 find() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find (
    const basic_string<_CharT, _Traits, _Alloc > & __str,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a string.

## Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5196 of file `basic_string.h`.

## 5.637.3.45 find() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find (
    const _CharT * __s,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a C string.

## Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5211 of file `basic_string.h`.

**5.637.3.46** `find()` [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
    _CharT __c,
    size_type __pos = 0 ) const [noexcept]
```

Find position of a character.

**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1224 of file `basic_string.tcc`.

**5.637.3.47** `find_first_not_of()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5504 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_first_not_of()`.

## 5.637.3.48 find\_first\_not\_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const [noexcept]
```

Find position of a character not in C substring.

## Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

## Returns

Index of first occurrence.

Starting from \_\_pos, searches forward for a character not contained in the first \_\_n characters of \_\_s within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1319 of file basic\_string.tcc.

## 5.637.3.49 find\_first\_not\_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_not_of (
    const _CharT * __s,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character not in C string.

## Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

## Returns

Index of first occurrence.

Starting from \_\_pos, searches forward for a character not contained in \_\_s within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 5535 of file basic\_string.h.

#### 5.637.3.50 find\_first\_not\_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
    _CharT __c,
    size_type __pos = 0 ) const [noexcept]
```

Find position of a different character.

##### Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1332 of file basic\_string.tcc.

#### 5.637.3.51 find\_first\_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character of string.

##### Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5337 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_first_of()`.

#### 5.637.3.52 find\_first\_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const [noexcept]
```

Find position of a character of C substring.

##### Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1281 of file `basic_string.tcc`.

#### 5.637.3.53 find\_first\_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_of (
    const _CharT * __s,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character of C string.

##### Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5368 of file `basic_string.h`.

**5.637.3.54 find\_first\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
    _CharT __c,
    size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

Definition at line 5388 of file `basic_string.h`.

**5.637.3.55 find\_last\_not\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character not in string.

**Parameters**

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5586 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

**5.637.3.56 find\_last\_not\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const [noexcept]
```

Find last position of a character not in C substring.

**Parameters**

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1343 of file `basic_string.tcc`.

**5.637.3.57 find\_last\_not\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
    const _CharT * __s,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character not in C string.



**Parameters**

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5617 of file `basic_string.h`.

**5.637.3.58 find\_last\_not\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
    _CharT __c,
    size_type __pos = npos ) const [noexcept]
```

Find last position of a different character.

**Parameters**

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1365 of file `basic_string.tcc`.

**5.637.3.59 find\_last\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5421 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_of()`.

**5.637.3.60 find\_last\_of()** [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const [noexcept]
```

Find last position of a character of C substring.

**Parameters**

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1297 of file `basic_string.tcc`.

**5.637.3.61** `find_last_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    const _CharT * __s,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character of C string.

**Parameters**

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5452 of file `basic_string.h`.

**5.637.3.62** `find_last_of()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
    _CharT __c,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

Definition at line 5472 of file `basic_string.h`.

**5.637.3.63** front() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::front ( ) [inline]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 4045 of file basic\_string.h.

**5.637.3.64** front() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::front ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 4056 of file basic\_string.h.

**5.637.3.65** get\_allocator()

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string<_CharT, _Traits, _Alloc>::get_allocator ( ) const [inline],
[noexcept]
```

Return copy of allocator used to construct this string.

Definition at line 5166 of file basic\_string.h.

Referenced by std::basic\_string<char>::assign(), std::basic\_string<char>::basic\_string(), std::basic\_string<char>::clear(), std::wstring\_convert<\_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc>::from\_bytes(), std::basic\_string<char>::swap(), std::wstring\_convert<\_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc>::to\_bytes(), and std::basic\_string<char>::~~basic\_string().

**5.637.3.66** insert() [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::insert (
    iterator __p,
    size_type __n,
    _CharT __c ) [inline]
```

Insert multiple characters.

**Parameters**

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4423 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

**5.637.3.67 insert()** [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    _InputIterator __beg,
    _InputIterator __end ) [inline]
```

Insert a range of characters.

**Parameters**

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[__beg,__end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4440 of file `basic_string.h`.

## 5.637.3.68 insert() [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::insert (
    iterator __p,
    initializer_list<_CharT > __l ) [inline]
```

Insert an initializer\_list of characters.

## Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__l</code>	The initializer_list of characters to insert.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 4451 of file basic\_string.h.

## 5.637.3.69 insert() [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (
    size_type __pos1,
    const basic_string<_CharT, _Traits, _Alloc > & __str ) [inline]
```

Insert value of a string.

## Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is

thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4471 of file `basic_string.h`.

#### 5.637.3.70 `insert()` [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n = npos ) [inline]
```

Insert a substring.

##### Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in <code>str</code> to insert.
<code>__n</code>	Number of characters to insert.

##### Returns

Reference to this string.

##### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> .

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4493 of file `basic_string.h`.

#### 5.637.3.71 `insert()` [6/9]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    const _CharT * __s,
    size_type __n )
```

Insert a C substring.

## Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.
<code>__n</code>	The number of characters to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 810 of file `basic_string.tcc`.

## 5.637.3.72 insert() [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    const _CharT * __s ) [inline]
```

Insert a C string.

## Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.



Inserts the first  $n$  characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4534 of file `basic_string.h`.

#### 5.637.3.73 `insert()` [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
    size_type __pos,
    size_type __n,
    _CharT __c ) [inline]
```

Insert multiple characters.

##### Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

##### Returns

Reference to this string.

##### Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4557 of file `basic_string.h`.

#### 5.637.3.74 `insert()` [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
    iterator __p,
    _CharT __c ) [inline]
```

Insert one character.

## Parameters

$\_p$	Iterator referencing position in string to insert at.
$\_c$	The character to insert.

## Returns

Iterator referencing newly inserted char.

## Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4575 of file `basic_string.h`.

## 5.637.3.75 length()

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::length ( ) const [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3845 of file `basic_string.h`.

Referenced by `std::collate<_CharT>::do_compare()`, and `std::collate<_CharT>::do_transform()`.

## 5.637.3.76 max\_size()

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size ( ) const [inline], [noexcept]
```

Returns the `size()` of the largest possible string.

Definition at line 3850 of file `basic_string.h`.

Referenced by `std::getline()`.

## 5.637.3.77 operator+=( ) [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
    const basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]
```

Append a string to this string.

**Parameters**

<code>__str</code>	The string to append.
--------------------	-----------------------

**Returns**

Reference to this string.

Definition at line 4092 of file `basic_string.h`.

**5.637.3.78 operator+=()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
    const _CharT * __s ) [inline]
```

Append a C string.

**Parameters**

<code>__s</code>	The C string to append.
------------------	-------------------------

**Returns**

Reference to this string.

Definition at line 4101 of file `basic_string.h`.

**5.637.3.79 operator+=()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
    _CharT __c ) [inline]
```

Append a character.

**Parameters**

<code>__c</code>	The character to append.
------------------	--------------------------

**Returns**

Reference to this string.

Definition at line 4110 of file basic\_string.h.

**5.637.3.80 operator+=( )** [ 4 / 4 ]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator+= (
    initializer_list<_CharT > __l ) [inline]
```

Append an initializer\_list of characters.

**Parameters**

↩	The initializer_list of characters to be appended.
↩	
↩	
↩	
/	

**Returns**

Reference to this string.

Definition at line 4123 of file basic\_string.h.

**5.637.3.81 operator=( )** [ 1 / 5 ]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator= (
    const basic_string<_CharT, _Traits, _Alloc > & __str ) [inline]
```

Assign the value of *str* to this string.

**Parameters**

__str	Source string.
-------	----------------


Definition at line 3651 of file basic\_string.h.

**5.637.3.82 operator=()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    const _CharT * __s ) [inline]
```

Copy contents of *s* into this string.

**Parameters**

 __s	Source null-terminated string.
--	--------------------------------


Definition at line 3659 of file basic\_string.h.

**5.637.3.83 operator=()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    _CharT __c ) [inline]
```

Set value to string of length 1.

**Parameters**

 __c	Source character.
--	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 3670 of file basic\_string.h.

**5.637.3.84 operator=()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (
    basic_string< _CharT, _Traits, _Alloc > && __str ) [inline]
```

Move assign the value of *str* to this string.

**Parameters**

__str	Source string.
-------	----------------

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 3686 of file basic\_string.h.

#### 5.637.3.85 operator=() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator= (
    initializer_list<_CharT > __l ) [inline]
```

Set value to string constructed from initializer list.

##### Parameters

↩	std::initializer_list.
↩	
↩	
↩	
↩	

Definition at line 3698 of file basic\_string.h.

#### 5.637.3.86 operator[]() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc >::operator[] (
    size_type __pos ) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

##### Parameters

__pos	The index of the character to access.
-------	---------------------------------------

##### Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out\_of\_range lookups are not defined. (For checked lookups see at().)

Definition at line 3967 of file basic\_string.h.

**5.637.3.87** `operator[]()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic\_string< _CharT, _Traits, _Alloc >::operator[] (
    size_type __pos ) [inline]
```

Subscript access to the data contained in the string.

**Parameters**

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 3984 of file `basic_string.h`.

**5.637.3.88** `pop_back()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic\_string< _CharT, _Traits, _Alloc >::pop_back ( ) [inline]
```

Remove the last character.

The string must be non-empty.

Definition at line 4681 of file `basic_string.h`.

**5.637.3.89** `push_back()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic\_string< _CharT, _Traits, _Alloc >::push_back (
    _CharT __c ) [inline]
```

Append a single character.

**Parameters**

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 4257 of file basic\_string.h.

Referenced by std::basic\_string<char>::operator+=( ).

#### 5.637.3.90 rbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin ( ) [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 3768 of file basic\_string.h.

#### 5.637.3.91 rbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 3777 of file basic\_string.h.

#### 5.637.3.92 rend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend ( ) [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 3786 of file basic\_string.h.

#### 5.637.3.93 rend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 3795 of file basic\_string.h.

#### 5.637.3.94 replace() [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (
    size_type __pos,
    size_type __n,
    const basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]
```

Replace characters with value from another string.



**Parameters**

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::out_of_range</code>	If <i>pos</i> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4706 of file `basic_string.h`.

Referenced by `std::basic_string< char >::append()`, `std::basic_string< char >::assign()`, `std::basic_string< char >::insert()`, and `std::basic_string< char >::replace()`.

**5.637.3.95 replace()** [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos1,
    size_type __n1,
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos2,
    size_type __n2 = npos ) [inline]
```

Replace characters with value from another string.

**Parameters**

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of <i>str</i> to use.
<code>__n2</code>	Number of characters from <i>str</i> to use.

**Returns**

Reference to this string.

**Exceptions**

<i>std::out_of_range</i>	If <i>__pos1</i> > size() or <i>__pos2</i> > __str.size().
<i>std::length_error</i>	If new length exceeds max_size().

Removes the characters in the range [*\_\_pos1*, *\_\_pos1* + *n*) from this string. In place, the value of *\_\_str* is inserted. If *\_\_pos* is beyond end of string, out\_of\_range is thrown. If the length of the result exceeds max\_size(), length\_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4728 of file basic\_string.h.

**5.637.3.96 replace()** [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc > & std::basic_string<_CharT, _Traits, _Alloc >::replace
(
    size_type __pos,
    size_type __n1,
    const _CharT * __s,
    size_type __n2 )
```

Replace characters with value of a C substring.

**Parameters**

<i>__pos</i>	Index of first character to replace.
<i>__n1</i>	Number of characters to be replaced.
<i>__s</i>	C string to insert.
<i>__n2</i>	Number of characters from <i>s</i> to use.

**Returns**

Reference to this string.

**Exceptions**

<i>std::out_of_range</i>	If <i>pos1</i> > size().
<i>std::length_error</i>	If new length exceeds max_size().

Removes the characters in the range [*\_\_pos*, *\_\_pos* + *\_\_n1*) from this string. In place, the first *\_\_n2* characters of *\_\_s* are inserted, or all of *\_\_s* if *\_\_n2* is too large. If *\_\_pos* is beyond end of string, out\_of\_range is thrown. If the length of

result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 864 of file `basic_string.tcc`.

#### 5.637.3.97 `replace()` [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n1,
    const _CharT * __s ) [inline]
```

Replace characters with value of a C string.

##### Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

##### Returns

Reference to this string.

##### Exceptions

<code>std::out_of_range</code>	If <code>pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4773 of file `basic_string.h`.

#### 5.637.3.98 `replace()` [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
    size_type __pos,
    size_type __n1,
    size_type __n2,
    _CharT __c ) [inline]
```

Replace characters with multiple characters.

## Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4797 of file `basic_string.h`.

## 5.637.3.99 replace() [6/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    const basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]
```

Replace range of characters with string.

## Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4815 of file `basic_string.h`.

5.637.3.100 `replace()` [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s,
    size_type __n ) [inline]
```

Replace range of characters with C substring.

## Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

## Returns

Reference to this string.

## Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.




Definition at line 4834 of file `basic_string.h`.

**5.637.3.101 replace()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    const _CharT * __s ) [inline]
```

Replace range of characters with C string.

**Parameters**

 __i1	Iterator referencing start of range to replace.
 __i2	Iterator referencing end of range to replace.
 __s	C string value to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range [`__i1`,`__i2`). In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.


Definition at line 4855 of file `basic_string.h`.

**5.637.3.102 replace()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    size_type __n,
    _CharT __c ) [inline]
```

Replace range of characters with multiple characters.

**Parameters**

 __i1	Iterator referencing start of range to replace.
---	---

**Parameters**

<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

**Returns**

Reference to this string.

**Exceptions**

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4876 of file `basic_string.h`.

**5.637.3.103 `replace()`** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
    iterator __i1,
    iterator __i2,
    _InputIterator __k1,
    _InputIterator __k2 ) [inline]
```

Replace range of characters with range.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4900 of file `basic_string.h`.

**5.637.3.104 replace()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (
    iterator __i1,
    iterator __i2,
    initializer_list<_CharT> __l ) [inline]
```

Replace range of characters with `initializer_list`.

**Parameters**

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4969 of file `basic_string.h`.



**5.637.3.105 reserve()**

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
    size_type __res_arg = 0 )
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

**Exceptions**

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 952 of file `basic_string.tcc`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::tr2::operator>>()`, `std::basic_string< char >::push_back()`, and `std::basic_string< char >::shrink_to_fit()`.

**5.637.3.106 resize()** [1/2]

```
template<typename _CharT, typename _Traits , typename _Alloc >
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
    size_type __n,
    _CharT __c )
```

Resizes the string to the specified number of characters.

**Parameters**

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 1090 of file basic\_string.tcc.


Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_string< char >::resize().

#### 5.637.3.107 resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
    size_type __n ) [inline]
```

Resizes the string to the specified number of characters.

##### Parameters

 <code>__n</code>	Number of characters the string should contain.
--	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 3877 of file basic\_string.h.

#### 5.637.3.108 rfind() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
    const basic_string< _CharT, _Traits, _Alloc > & __str,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a string.

##### Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

##### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5258 of file basic\_string.h.

Referenced by `std::basic_string< char >::find_last_of()`, and `std::basic_string< char >::rfind()`.

#### 5.637.3.109 `rfind()` [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
    const _CharT * __s,
    size_type __pos,
    size_type __n ) const [noexcept]
```

Find last position of a C substring.

##### Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

##### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1242 of file `basic_string.tcc`.

#### 5.637.3.110 `rfind()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
    const _CharT * __s,
    size_type __pos = npos ) const [inline], [noexcept]
```

Find last position of a C string.

##### Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5289 of file `basic_string.h`.

**5.637.3.111 rfind()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc
>::rfind (
    _CharT __c,
    size_type __pos = npos ) const [noexcept]
```

Find last position of a character.

**Parameters**

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1264 of file `basic_string.tcc`.

**5.637.3.112 shrink\_to\_fit()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::shrink_to_fit ( ) [inline], [noexcept]
```

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 3883 of file `basic_string.h`.

**5.637.3.113 size()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3839 of file basic\_string.h.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, `std::basic_string< char >::assign()`, `std::basic_string< char >::at()`, `std::basic_string< char >::cend()`, `std::basic_string< char >::compare()`, `std::basic_string< char >::empty()`, `std::basic_string< char >::end()`, `std::match_results< _Bi_iter >::format()`, `std::wstring_convert< _Codecv, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes()`, `std::operator<()`, `std::operator==()`, `std::basic_string< char >::push_back()`, `std::basic_string< char >::shrink_to_fit()`, `std::wstring_convert< _Codecv, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes()`, and `std::regex_traits< _CharType >::transform()`.

**5.637.3.114 substr()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
    size_type __pos = 0,
    size_type __n = npos ) const [inline]
```

Get a substring.

**Parameters**

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

**Returns**

The new string.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos &gt; size()</code> .
--------------------------------	-------------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 5669 of file basic\_string.h.

## 5.637.3.115 swap()

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
    basic_string< _CharT, _Traits, _Alloc > & __s )
```

Swap contents with another string.

## Parameters

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 969 of file `basic_string.tcc`.

Referenced by `std::basic_string< char >::assign()`, and `std::basic_string< char >::operator=()`.

## 5.637.4 Member Data Documentation

## 5.637.4.1 npos

```
template<typename _CharT, typename _Traits, typename _Alloc>
const basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _
_Alloc >::npos [static]
```

Value returned by various member functions when they fail.

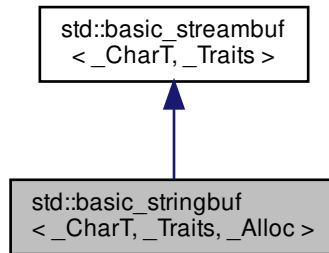
Definition at line 3310 of file `basic_string.h`.

The documentation for this class was generated from the following files:

- [basic\\_string.h](#)
- [basic\\_string.tcc](#)

## 5.638 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

Inheritance diagram for std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >:



### Public Types

- typedef \_\_string\_type::size\_type **\_\_size\_type**
- typedef [basic\\_streambuf](#)< char\_type, traits\_type > **\_\_streambuf\_type**
- typedef [basic\\_string](#)< char\_type, \_Traits, \_Alloc > **\_\_string\_type**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

### Public Member Functions

- [basic\\_stringbuf](#) (ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [basic\\_stringbuf](#) (const \_\_string\_type &\_\_str, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **basic\_stringbuf** (const [basic\\_stringbuf](#) &)=delete
- **basic\_stringbuf** ([basic\\_stringbuf](#) &&\_\_rhs)
- [locale](#) [getloc](#) () const
- [streamsize](#) [in\\_avail](#) ()
- [basic\\_stringbuf](#) & **operator=** (const [basic\\_stringbuf](#) &)=delete
- [basic\\_stringbuf](#) & **operator=** ([basic\\_stringbuf](#) &&\_\_rhs)
- [locale](#) [pubimbue](#) (const [locale](#) &\_\_loc)
- int\_type [sbumpc](#) ()
- int\_type [sgetc](#) ()
- [streamsize](#) [sgetn](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- int\_type [snextc](#) ()
- int\_type [sputbackc](#) (char\_type \_\_c)
- int\_type [sputc](#) (char\_type \_\_c)

- [streamsize sputn](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_string\\_type str](#) () const
- void [str](#) (const [\\_\\_string\\_type](#) & \_\_s)
- int\_type [sungetc](#) ()
- void [swap](#) ([basic\\_stringbuf](#) & \_\_rhs)
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- int [pubsync](#) ()

#### Protected Member Functions

- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
- void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
- void [\\_M\\_pbump](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend, off\_type \_\_off)
- void [\\_M\\_stringbuf\\_init](#) ([ios\\_base::openmode](#) \_\_mode)
- void [\\_M\\_sync](#) (char\_type \* \_\_base, \_\_size\_type \_\_i, \_\_size\_type \_\_o)
- void [\\_M\\_update\\_egptr](#) ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) & \_\_loc \_\_attribute\_\_((\_\_unused\_\_)))
- virtual int\_type [overflow](#) (int\_type \_\_c=[traits\\_type::eof](#)())
- virtual int\_type [overflow](#) (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=[traits\\_type::eof](#)())
- virtual int\_type [pbackfail](#) (int\_type \_\_c=[traits\\_type::eof](#)())
- virtual int\_type [pbackfail](#) (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=[traits\\_type::eof](#)())
- void [pbump](#) (int \_\_n)
- virtual pos\_type [seekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual pos\_type [seekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual [streamsize showmanyc](#) ()
- void [swap](#) ([basic\\_streambuf](#) & \_\_sb)
- virtual int [sync](#) ()
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- virtual [streamsize xsgetn](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- virtual [streamsize xsputn](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const
- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const



## Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [ios\\_base::openmode](#) [\\_M\\_mode](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)
- [\\_\\_string\\_type](#) [\\_M\\_string](#)

### 5.638.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringbuf< _CharT, _Traits, _Alloc >
```

The actual work of input and output (for `std::string`).

#### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.)

For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 96 of file `iosfwd`.

### 5.638.2 Constructor & Destructor Documentation

#### 5.638.2.1 `basic_stringbuf()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [explicit]
```

Starts with an empty string buffer.

## Parameters

<code>__mode</code>	Whether the buffer can read, or write, or both.
---------------------	---

The default constructor initializes the parent class using its own default ctor.

Definition at line 100 of file sstream.

## 5.638.2.2 basic\_stringbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
    const __string_type & __str,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [explicit]
```

Starts with an existing string buffer.

## Parameters

<code>__str</code>	A string to copy as a starting buffer.
<code>__mode</code>	Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

Definition at line 113 of file sstream.

## 5.638.3 Member Function Documentation

## 5.638.3.1 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file streambuf.

### 5.638.3.2 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

### 5.638.3.3 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

### 5.638.3.4 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the read position.

#### Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

#### 5.638.3.5 getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

#### 5.638.3.6 gptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

#### 5.638.3.7 imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
    const locale &__loc __attribute__((__unused__)) ) [inline], [protected], [virtual],
[inherited]
```

Changes translations.

**Parameters**

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Definition at line 583 of file streambuf.

**5.638.3.8 in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file streambuf.

**5.638.3.9 overflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
    int_type __c __attribute__((__unused__)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

**Returns**

eof() to indicate failure, something else (usually \_\_c, or not\_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if \_\_c is not eof().

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns eof().


Definition at line 775 of file streambuf.

**5.638.3.10 pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

 __c	The character to be inserted back into the sequence.
--	--

**Returns**

eof() on failure, *some other value* on success

**Postcondition**

The constraints of gptr(), eback(), and pptr() are the same as for underflow().

**Note**

Base class version does nothing, returns eof().

Definition at line 731 of file streambuf.

### 5.638.3.11 pbase()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pbase ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 536 of file streambuf.

### 5.638.3.12 pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the write position.

#### Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

### 5.638.3.13 pptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

#### 5.638.3.14 pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_stringbuf< _CharT, _Traits >::pubimbue (
    const locale & __loc ) [inline], [inherited]
```

Entry point for imbue().

##### Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

##### Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

#### 5.638.3.15 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_stringbuf< _CharT, _Traits >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

##### Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for ios_base::seekdir.
<code>__mode</code>	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

#### 5.638.3.16 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_stringbuf< _CharT, _Traits >::pubseekpos (
```



```
pos_type __sp,  
ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual `seekpos` function.

Definition at line 270 of file `streambuf`.

### 5.638.3.17 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

### 5.638.3.18 pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

### 5.638.3.19 sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

**5.638.3.20 seekoff()**

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out ) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 168 of file `sstream.tcc`.

References `std::ios_base::cur`, `std::ios_base::end`, `std::ios_base::in`, and `std::ios_base::out`.

**5.638.3.21 seekpos()**

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out ) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 216 of file `sstream.tcc`.

References `std::ios_base::in`, and `std::ios_base::out`.

**5.638.3.22 setbuf()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (
    char_type * __s,
    streamsize __n ) [inline], [protected], [virtual]
```

Manipulates the buffer.

## Parameters

<code>__s</code>	Pointer to a buffer area.
<code>__n</code>	Size of <code>__s</code> .

## Returns

`this`

If no buffer has already been created, and both `__s` and `__n` are non-zero, then `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 244 of file `sstream`.

## 5.638.3.23 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected], [inherited]
```

Setting the three read area pointers.

## Parameters

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

## Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file `streambuf`.

**5.638.3.24 setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
    char_type * __pbeg,
    char_type * __pend ) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

**5.638.3.25 sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file streambuf.

**5.638.3.26 sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for `xsggetn`.

## Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

## 5.638.3.27 showmanyc()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc ( ) [inline],
[protected], [virtual]
```

Investigating the data available.

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 212 of file `sstream`.

## 5.638.3.28 snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.


Definition at line 305 of file `streambuf`.

**5.638.3.29 sputback()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputback (
    char_type __c ) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

 <code>__c</code>	The character to push back.
--	-----------------------------

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.


Definition at line 379 of file `streambuf`.

**5.638.3.30 sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
    char_type __c ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

 <code>__c</code>	A character to output.
--	------------------------

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**5.638.3.31 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_stringbuf< _CharT, _Traits >::sputn (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**5.638.3.32 str()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( ) const [inline]
```

Copying out the string buffer.

**Returns**

A copy of one of the underlying sequences.

*If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence.* [27.7.1.2]/1

Definition at line 167 of file `sstream`.

**5.638.3.33 str()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
    const __string_type & __s ) [inline]
```

Setting a new buffer.



**Parameters**

<code>_s</code>	The string to use as a new sequence.
-----------------	--------------------------------------

Deallocates any previous stored sequence, then copies *s* to use as a new one.

Definition at line 191 of file `sstream`.

**5.638.3.34 `sungetc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**5.638.3.35 `sync()`**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync (
    void ) [inline], [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, and `__gnu_cxx::stdio_sync_filebuf< _CharT`

Definition at line 634 of file `streambuf`.

## 5.638.3.36 uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

## 5.638.3.37 underflow()

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::underflow ( ) [protected], [virtual]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 150 of file `sstream.tcc`.

References `std::ios_base::in`.

## 5.638.3.38 xsggetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_streambuf< _CharT, _Traits >::xsggetn (
    char_type * __s,
    streamsize __n ) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 46 of file `streambuf.tcc`.

**5.638.3.39 xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
    const char_type * __s,
    streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 80 of file `streambuf.tcc`.

#### 5.638.4 Member Data Documentation

##### 5.638.4.1 \_M\_buf\_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_stringbuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file streambuf.

##### 5.638.4.2 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_stringbuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

##### 5.638.4.3 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_stringbuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

##### 5.638.4.4 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_stringbuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 5.638.4.5 `_M_mode`

```
template<typename _CharT, typename _Traits, typename _Alloc>
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 85 of file sstream.

#### 5.638.4.6 `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 5.638.4.7 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 5.638.4.8 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

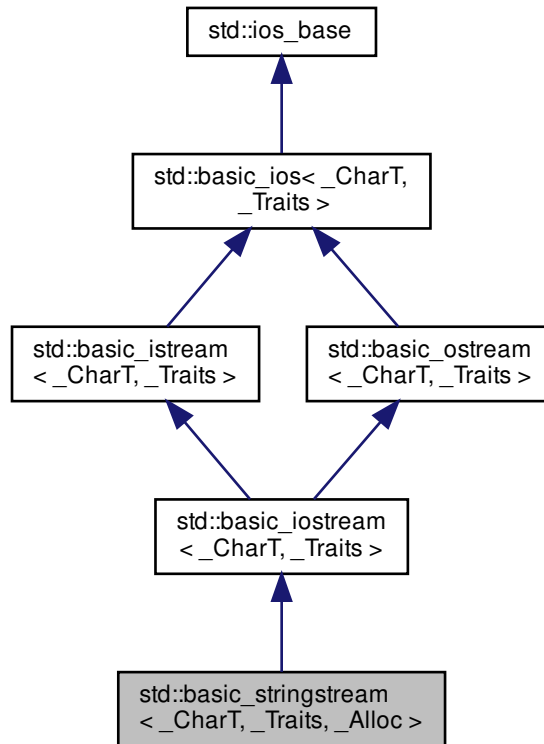
Definition at line 196 of file streambuf.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)
- [sstream.tcc](#)

## 5.639 std::basic\_stringstream&lt; \_CharT, \_Traits, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`

- typedef `_Alloc` **allocator\_type**
  - typedef `_CharT` **char\_type**
  - enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
  - typedef void(\* `event_callback`) (`event` \_\_e, `ios_base` & \_\_b, int \_\_i)
  - typedef `_ios_Fmtflags` **fmtflags**
  - typedef `traits_type::int_type` **int\_type**
  - typedef int **io\_state**
  - typedef `_ios_istate` **istate**
  - typedef `traits_type::off_type` **off\_type**
  - typedef int **open\_mode**
  - typedef `_ios_Openmode` **openmode**
  - typedef `traits_type::pos_type` **pos\_type**
  - typedef int **seek\_dir**
  - typedef `_ios_Seekdir` **seekdir**
  - typedef `std::streamoff` **streamoff**
  - typedef `std::streampos` **streampos**
  - typedef `_Traits` **traits\_type**
- 
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` **\_\_num\_put\_type**

#### Public Member Functions

- `basic_stringstream` (`ios_base::openmode` \_\_m=`ios_base::out|ios_base::in`)
- `basic_stringstream` (const `__string_type` & \_\_str, `ios_base::openmode` \_\_m=`ios_base::out|ios_base::in`)
- **basic\_stringstream** (const `basic_stringstream` &)=delete
- **basic\_stringstream** (`basic_stringstream` && \_\_rhs)
- `~basic_stringstream` ()
- template<typename `_ValueT` >  
`basic_istream`< `_CharT`, `_Traits` > & **\_M\_extract** (`_ValueT` & \_\_v)
- const `locale` & **\_M\_getloc** () const
- template<typename `_ValueT` >  
`basic_ostream`< `_CharT`, `_Traits` > & **\_M\_insert** (`_ValueT` \_\_v)
- void **\_M\_setstate** (`istate` \_\_state)
- bool **bad** () const
- void **clear** (`istate` \_\_state=`goodbit`)
- `basic_ios` & **copyfmt** (const `basic_ios` & \_\_rhs)
- bool **eof** () const
- `istate` **exceptions** () const
- void **exceptions** (`istate` \_\_except)
- bool **fail** () const
- `char_type` **fill** () const
- `char_type` **fill** (`char_type` \_\_ch)
- `fmtflags` **flags** () const
- `fmtflags` **flags** (`fmtflags` \_\_fmtfl)
- `__ostream_type` & **flush** ()
- `streamsize` **gcount** () const

- template<>  
basic\_istream< char > & [getline](#) (char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim)
  - template<>  
basic\_istream< wchar\_t > & [getline](#) (char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim)
  - locale [getloc](#) () const
  - bool [good](#) () const
  - template<>  
basic\_istream< char > & [ignore](#) (streamsize \_\_n)
  - template<>  
basic\_istream< char > & [ignore](#) (streamsize \_\_n, int\_type \_\_delim)
  - template<>  
basic\_istream< wchar\_t > & [ignore](#) (streamsize \_\_n)
  - template<>  
basic\_istream< wchar\_t > & [ignore](#) (streamsize \_\_n, int\_type \_\_delim)
  - locale [imbue](#) (const locale &\_\_loc)
  - long & [iword](#) (int \_\_ix)
  - char [narrow](#) (char\_type \_\_c, char \_\_default) const
  - \_\_ostream\_type & [operator<<](#) (const void \*\_\_p)
  - \_\_ostream\_type & [operator<<](#) (\_\_streambuf\_type \*\_\_sb)
  - basic\_stringstream & [operator=](#) (const basic\_stringstream &)=delete
  - basic\_stringstream & [operator=](#) (basic\_stringstream &&\_\_rhs)
  - \_\_istream\_type & [operator>>](#) (void \*&\_\_p)
  - \_\_istream\_type & [operator>>](#) (\_\_streambuf\_type \*\_\_sb)
  - streamsize [precision](#) () const
  - streamsize [precision](#) (streamsize \_\_prec)
  - void \*& [pword](#) (int \_\_ix)
  - basic\_streambuf< \_CharT, \_Traits > \* [rdbuf](#) (basic\_streambuf< \_CharT, \_Traits > \*\_\_sb)
  - \_\_stringbuf\_type \* [rdbuf](#) () const
  - iostate [rdstate](#) () const
  - void [register\\_callback](#) (event\_callback \_\_fn, int \_\_index)
  - \_\_ostream\_type & [seekp](#) (pos\_type)
  - \_\_ostream\_type & [seekp](#) (off\_type, ios\_base::seekdir)
  - fmtflags [setf](#) (fmtflags \_\_fmtfl)
  - fmtflags [setf](#) (fmtflags \_\_fmtfl, fmtflags \_\_mask)
  - void [setstate](#) (iostate \_\_state)
  - \_\_string\_type str () const
  - void [str](#) (const \_\_string\_type &\_\_s)
  - void [swap](#) (basic\_stringstream &\_\_rhs)
  - pos\_type [tellp](#) ()
  - basic\_ostream< \_CharT, \_Traits > \* [tie](#) () const
  - basic\_ostream< \_CharT, \_Traits > \* [tie](#) (basic\_ostream< \_CharT, \_Traits > \*\_\_tiestr)
  - void [unsetf](#) (fmtflags \_\_mask)
  - char\_type [widen](#) (char \_\_c) const
  - streamsize [width](#) () const
  - streamsize [width](#) (streamsize \_\_wide)
- 
- \_\_istream\_type & [operator>>](#) (\_\_istream\_type &(\*\_\_pf)(\_\_istream\_type &))
  - \_\_istream\_type & [operator>>](#) (\_\_ios\_type &(\*\_\_pf)(\_\_ios\_type &))



- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
  - `__istream_type & operator>> (short &__n)`
  - `__istream_type & operator>> (unsigned short &__n)`
  - `__istream_type & operator>> (int &__n)`
  - `__istream_type & operator>> (unsigned int &__n)`
  - `__istream_type & operator>> (long &__n)`
  - `__istream_type & operator>> (unsigned long &__n)`
  - `__istream_type & operator>> (long long &__n)`
  - `__istream_type & operator>> (unsigned long long &__n)`
- 
- `__istream_type & operator>> (float &__f)`
  - `__istream_type & operator>> (double &__f)`
  - `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`

- [\\_\\_istream\\_type](#) & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - bool [operator!](#) () const
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iostate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { **[\\_S\\_local\\_word\\_size](#)** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT>  
  [\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT>  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_iostream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)<\_CharT, \_Traits> \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 5.639.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringstream<_CharT, _Traits, _Alloc>
```

Controlling input and output for std::string.

### Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 108 of file `iosfwd`.

## 5.639.2 Member Typedef Documentation

### 5.639.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

### 5.639.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 506 of file `ios_base.h`.

## 5.639.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 323 of file ios\_base.h.

#### 5.639.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- *badbit*
- *eofbit*
- *failbit*
- *goodbit*

Definition at line 398 of file *ios\_base.h*.

#### 5.639.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 429 of file *ios\_base.h*.

## 5.639.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

## 5.639.3 Member Enumeration Documentation

## 5.639.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.639.4 Constructor &amp; Destructor Documentation

## 5.639.4.1 basic\_stringstream() [1/2]

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
    ios_base::openmode __m = ios_base::out | ios_base::in ) [inline], [explicit]
```

Default constructor starts with an empty string buffer.

## Parameters

<code>__m</code>	Whether the buffer can read, or write, or both.
------------------	---



Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 704 of file `sstream`.

#### 5.639.4.2 `basic_stringstream()` [2/2]

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
    const __string_type & __str,
    ios_base::openmode __m = ios_base::out | ios_base::in ) [inline], [explicit]
```

Starts with an existing string buffer.

##### Parameters

<code>__str</code>	A string to copy as a starting buffer.
<code>__m</code>	Whether the buffer can read, or write, or both.

Initializes `sb` using `__str` and `__m`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 720 of file `sstream`.

#### 5.639.4.3 `~basic_stringstream()`

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::~~basic_stringstream ( ) [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 731 of file `sstream`.

### 5.639.5 Member Function Documentation

## 5.639.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]
```

Locale access.

## Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 776 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

## 5.639.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic\_ostream< _CharT, _Traits >::_M_write (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Core write functionality, without sentry.

## Parameters

<a href="#">_s</a>	The array to insert.
<a href="#">_n</a>	Maximum number of characters to insert.

Definition at line 311 of file ostream.

## 5.639.5.3 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic\_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**5.639.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
    iostate __state = goodbit ) [inherited]
```

[Re]sets the error state.

**Parameters**

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

**5.639.5.5 copyfmt()**

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
    const basic_ios< _CharT, _Traits > & __rhs ) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the pword

and iword arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

#### 5.639.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof ( ) const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file `basic_ios.h`.

#### 5.639.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`.

#### 5.639.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
    iostate __except ) [inline], [inherited]
```

Throwing exceptions on errors.

### Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
    std::ifstream f ("/etc/motd");
    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);
    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

### 5.639.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]
```

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, `std::basic_ios< char, _Traits >::operator!()`, and `std::regex_traits< _CharType >::value()`.

### 5.639.5.10 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]
```

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

## 5.639.5.11 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
    char_type __ch ) [inline], [inherited]
```

Sets a new *empty* character.

## Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

## Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

## 5.639.5.12 flags() [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline], [inherited]
```

Access to format flags.

## Returns

The format control flags for both input and output.

Definition at line 621 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

## 5.639.5.13 flags() [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file `ios_base.h`.

**5.639.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ( ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( )->pubsync ( )`, and if that returns -1, sets badbit.

Definition at line 211 of file `ostream.tcc`.

**5.639.5.15 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file `istream`.

**5.639.5.16** get() [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

**5.639.5.17** get() [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type & __c ) [inherited]
```

Simple extraction.

**Parameters**

<code>__c</code>	The character in which to store data.
------------------	---------------------------------------

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file istream.tcc.

**5.639.5.18** get() [3/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Simple multiple-character extraction.



**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>__s</code> .
<code>__delim</code>	A "stop" character.

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file `istream.tcc`.

**5.639.5.19 `get()`** [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

<code>__s</code>	Pointer to an array.
<code>__n</code>	Maximum number of characters to store in <code>s</code> .

## Returns

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

5.639.5.20 `get()` [5/6]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb,
    char_type __delim ) [inherited]
```

Extraction into another streambuf.

## Parameters

<code>__sb</code>	A streambuf in which to store data.
<code>__delim</code>	A "stop" character.

## Returns

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

5.639.5.21 `get()` [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
    __streambuf_type & __sb ) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

<code>__sb</code>	A streambuf in which to store data.
-------------------	-------------------------------------

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

**5.639.5.22 getline()** [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim )    [inherited]
```

String extraction.

**Parameters**

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.
<code>__delim</code>	A "stop" character.

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file istream.tcc.

## 5.639.5.23 getline() [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::getline (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

String extraction.

## Parameters

<code>__s</code>	A character array in which to store the data.
<code>__n</code>	Maximum number of characters to extract.

## Returns

\*this

Returns `getline(__s, __n, widen("\n"))`.

Definition at line 427 of file istream.

## 5.639.5.24 getline() [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
    char_type * __s,
    streamsize __n,
    char_type __delim ) [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

## 5.639.5.25 getloc()

```
locale std::ios_base::getloc ( ) const [inline], [inherited]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outiter>::do_put()`, and `std::ws()`.

**5.639.5.26** `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.639.5.27** `ignore()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n,
    int_type __delim ) [inherited]
```

Discarding characters.

**Parameters**

<code>__n</code>	Number of characters to discard.
<code>__delim</code>	A "stop" character.

**Returns**

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**5.639.5.28 ignore()** [2/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    streamsize __n ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 501 of file istream.tcc.

**5.639.5.29 ignore()** [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
    void ) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**5.639.5.30 imbue()**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
    const locale & __loc ) [inherited]
```

Moves to a new locale.

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**5.639.5.31 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
    basic_streambuf< _CharT, _Traits > * __sb ) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**5.639.5.32 iword()**

```
long& std::ios_base::iword (
    int __ix ) [inline], [inherited]
```

Access to integer array.

**Parameters**

<code>__ix</code>	Index into the array.
-------------------	-----------------------

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file ios\_base.h.

#### 5.639.5.33 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 430 of file basic\_ios.h.

#### 5.639.5.34 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool ( ) const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file basic\_ios.h.



**5.639.5.35 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**5.639.5.36 operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ostream_type &(*) (__ostream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**5.639.5.37 operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**5.639.5.38 operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

**5.639.5.39 operator<<() [4/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

## 5.639.5.40 operator&lt;&lt;() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned long __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

<code>__n</code>	A variable of builtin integral type.
------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 170 of file ostream.

## 5.639.5.41 operator&lt;&lt;() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    bool __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 174 of file `ostream`.

**5.639.5.42 operator<<() [7/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
    short __n ) [inherited]
```

Integer arithmetic inserters.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

**5.639.5.43 operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned short __n ) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

## 5.639.5.44 operator&lt;&lt;() [9/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    int __n ) [inherited]
```

Integer arithmetic inserters.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 106 of file ostream.tcc.

## 5.639.5.45 operator&lt;&lt;() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    unsigned int __n ) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 192 of file `ostream`.

**5.639.5.46 operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long long __n )    [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

 <code>__n</code>	A variable of builtin integral type.
--	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

**5.639.5.47 operator<<() [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    unsigned long long __n )    [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

$\_n$	A variable of builtin integral type.
-------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

## 5.639.5.48 operator&lt;&lt;() [13/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

$\_f$	A variable of builtin floating point type.
-------	--

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

## 5.639.5.49 operator&lt;&lt;() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
    float __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**5.639.5.50 operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    long double __f ) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

↩	A variable of builtin floating point type.
↩	
↩	
↩	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**5.639.5.51 operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
    const void * __p ) [inline], [inherited]
```

Pointer arithmetic inserters.

## Parameters

<code>_p</code>	A variable of pointer type.
-----------------	-----------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

## 5.639.5.52 operator&lt;&lt;() [17/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
    __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

## Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.



**5.639.5.53 operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __istream_type &(*) (__istream_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

**5.639.5.54 operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    __ios_type &(*) (__ios_type &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

**5.639.5.55 operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    ios_base &(*) (ios_base &) __pf ) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

**5.639.5.56 operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    bool & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$ _n	A variable of builtin integral type.
--------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

## 5.639.5.57 operator&gt;&gt;() [5/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
    short & __n ) [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$ _n	A variable of builtin integral type.
--------------------	--------------------------------------

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 122 of file `istream.tcc`.

## 5.639.5.58 operator&gt;&gt;() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    unsigned short & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 175 of file `istream`.

**5.639.5.59 operator>>() [7/17]**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    int & __n ) [inherited]
```

Integer arithmetic extractors.

**Parameters**

	A variable of builtin integral type.
<code>__n</code>	

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**5.639.5.60 operator>>() [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned int & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

## 5.639.5.61 operator&gt;&gt;() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
<code>__n</code>	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

## 5.639.5.62 operator&gt;&gt;() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>__n</code>	A variable of builtin integral type.
----------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**5.639.5.63 `operator>>()` [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

$\leftarrow$ <code>__n</code>	A variable of builtin integral type.
----------------------------------	--------------------------------------

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**5.639.5.64 `operator>>()` [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    unsigned long long & __n ) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin integral type.
$\leftarrow n$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

## 5.639.5.65 operator&gt;&gt;() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    float & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

$\leftarrow$	A variable of builtin floating point type.
$\leftarrow$	
$\leftarrow$	
$\leftarrow$	
$\leftarrow f$	

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

## 5.639.5.66 operator&gt;&gt;() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
    double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**5.639.5.67 operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    long double & __f ) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

↵	A variable of builtin floating point type.
↵	
↵	
↵	
<i>f</i>	

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**5.639.5.68 operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
    void *& __p ) [inline], [inherited]
```

Basic arithmetic extractors.

## Parameters

<code>_p</code>	A variable of pointer type.
-----------------	-----------------------------

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

5.639.5.69 `operator>>()` [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
    __streambuf_type * __sb ) [inherited]
```

Extracting into another streambuf.

## Parameters

<code>__sb</code>	A pointer to a streambuf
-------------------	--------------------------

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.



**5.639.5.70 peek()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
    void ) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

**5.639.5.71 precision()** [1/2]

```
streamsize std::ios_base::precision ( ) const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.639.5.72 precision()** [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__prec</code>	The new precision value.
---------------------	--------------------------

**Returns**

The previous value of precision().


Definition at line 700 of file ios\_base.h.

**5.639.5.73 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
    char_type __c ) [inherited]
```

Simple insertion.

**Parameters**

	The character to insert.
<code>__c</code>	

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.


Definition at line 149 of file ostream.tcc.

**5.639.5.74 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
    char_type __c ) [inherited]
```

Unextracting a single character.

**Parameters**

	The character to push back into the input stream.
<code>__c</code>	

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**5.639.5.75 pword()**

```
void*& std::ios_base::pword (
    int __ix ) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

$\leftrightarrow$ __ix	Index into the array.
---------------------------	-----------------------

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file `ios_base.h`.

**5.639.5.76 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
    basic_streambuf< _CharT, _Traits > * __sb ) [inherited]
```

Changing the underlying buffer.

## Parameters

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;
foo.ios::rdbuf(p);           // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

5.639.5.77 `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_stringstream<_CharT, _Traits, _Alloc>::rdbuf ( ) const [inline]
```

Accessing the underlying buffer.

## Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 771 of file `sstream`.

5.639.5.78 `rdstate()`

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<char>::unset()`.

**5.639.5.79 read()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction without delimiters.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

\*this

If the stream state is `good()` , extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file istream.tcc.

**5.639.5.80 readsome()**

```
template<typename _CharT , typename _Traits >
streamsize std::basic_istream< _CharT, _Traits >::readsome (
    char_type * __s,
    streamsize __n ) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

<code>__s</code>	A character array.
<code>__n</code>	Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

**5.639.5.81 register\_callback()**

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index ) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.639.5.82 seekg()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    pos_type __pos ) [inherited]
```

Changing the current read position.

**Parameters**

<code>__pos</code>	A file position object.
--------------------	-------------------------

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**5.639.5.83 seekg()** [2/2]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current read position.

**Parameters**

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**5.639.5.84 seekp()** [1/2]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
    pos_type __pos ) [inherited]
```

Changing the current write position.

## Parameters

<code>__pos</code>	A file position object.
--------------------	-------------------------

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets `failbit`.

Definition at line 258 of file `ostream.tcc`.

## 5.639.5.85 seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
    off_type __off,
    ios_base::seekdir __dir ) [inherited]
```

Changing the current write position.

## Parameters

<code>__off</code>	A file offset object.
<code>__dir</code>	The direction in which to seek.

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

## 5.639.5.86 setf() [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline], [inherited]
```

Setting new format flags.

## Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------



**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**5.639.5.87 setf()** [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline], [inherited]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

**5.639.5.88 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
    iostate __state ) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by std::operator<<(), std::basic\_ostream<\_CharT, \_Traits>::sentry::sentry(), and std::ws().

#### 5.639.5.89 str() [1/2]

```
template<typename _CharT , typename _Traits , typename _Alloc >
__string_type std::basic_stringstream< _CharT, _Traits, _Alloc >::str ( ) const [inline]
```

Copying out the string buffer.

#### Returns

rdbuf() -> str()


Definition at line 779 of file sstream.

#### 5.639.5.90 str() [2/2]

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
    const __string_type & __s ) [inline]
```

Setting a new buffer.

#### Parameters

 <code>__s</code>	The string to use as a new sequence.
--	--------------------------------------

Calls rdbuf() -> str(s).

Definition at line 789 of file sstream.

#### 5.639.5.91 sync()

```
template<typename _CharT , typename _Traits >
int std::basic_istream< _CharT, _Traits >::sync (
    void ) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

**5.639.5.92 sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**5.639.5.93 tellg()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
    void ) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**5.639.5.94 tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp ( )
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**5.639.5.95 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( ) const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**5.639.5.96 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (
    basic_ostream<_CharT, _Traits> * __tiestr ) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**5.639.5.97 `unget()`**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
    void ) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**5.639.5.98 `unsetf()`**

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline], [inherited]
```

Clearing format flags.

## Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

## 5.639.5.99 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
    char __c ) const [inline], [inherited]
```

Widens characters.

## Parameters

<code>__c</code>	The character to widen.
------------------	-------------------------

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

## 5.639.5.100 width() [1/2]

```
streamsize std::ios_base::width ( ) const [inline], [inherited]
```

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

**5.639.5.101** `width()` [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline], [inherited]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

**5.639.5.102** `write()`

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
    const char_type * __s,
    streamsize __n ) [inherited]
```

Character string insertion.

**Parameters**

<code>__s</code>	The array to insert.
<code>__n</code>	Maximum number of characters to insert.

**Returns**

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

### 5.639.5.103 xalloc()

```
static int std::ios_base::xalloc ( ) throw ( ) [static], [inherited]
```

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.639.6 Member Data Documentation

#### 5.639.6.1 \_M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, and `std::basic_istream< char >::unget()`.

#### 5.639.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`.



#### 5.639.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 432 of file ios\_base.h.

#### 5.639.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 435 of file ios\_base.h.

#### 5.639.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 5.639.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 381 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.639.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file ios\_base.h.

#### 5.639.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Definition at line 440 of file ios\_base.h.

#### 5.639.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file ios\_base.h.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.639.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.639.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.

**5.639.6.12 end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.639.6.13 eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

**5.639.6.14 failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

**5.639.6.15 fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

#### 5.639.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file `ios_base.h`.

#### 5.639.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 413 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<char>::flush()`, `std::basic_istream<char>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_ostream<char>::operator<<()`, `std::basic_istream<char>::operator>>()`, `std::operator>>()`, `std::basic_istream<char>::peek()`, `std::basic_ostream<char>::put()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsome()`, `std::basic_istream<char>::seekg()`, `std::basic_ostream<char>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char>::sync()`, and `std::basic_istream<char>::unget()`.

#### 5.639.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::basic_ostream<char>::operator<<()`.

#### 5.639.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

#### 5.639.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file `ios_base.h`.

#### 5.639.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

#### 5.639.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::basic_ostream<char>::operator<<()`.

#### 5.639.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 446 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 5.639.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

#### 5.639.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

#### 5.639.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put().

#### 5.639.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.639.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

#### 5.639.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

#### 5.639.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 449 of file ios\_base.h.

#### 5.639.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

#### 5.639.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file ios\_base.h.

Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

## 5.640 std::bernoulli\_distribution Class Reference

### Classes

- struct [param\\_type](#)

## Public Types

- typedef bool [result\\_type](#)

## Public Member Functions

- [bernoulli\\_distribution](#) (double \_\_p=0.5)
- **bernoulli\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool [operator==](#) (const [bernoulli\\_distribution](#) &\_\_d1, const [bernoulli\\_distribution](#) &\_\_d2)

## 5.640.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Definition at line 3452 of file random.h.

## 5.640.2 Member Typedef Documentation

## 5.640.2.1 result\_type

```
typedef bool std::bernoulli_distribution::result_type
```

The type of the range of the distribution.

Definition at line 3456 of file random.h.



### 5.640.3 Constructor & Destructor Documentation

#### 5.640.3.1 bernoulli\_distribution()

```
std::bernoulli_distribution::bernoulli_distribution (
    double __p = 0.5 ) [inline], [explicit]
```

Constructs a Bernoulli distribution with likelihood  $p$ .

##### Parameters

$\leftrightarrow$ $p$	[IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$ .
--------------------------	---

Definition at line 3494 of file random.h.

### 5.640.4 Member Function Documentation

#### 5.640.4.1 max()

```
result_type std::bernoulli_distribution::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3544 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

#### 5.640.4.2 min()

```
result_type std::bernoulli_distribution::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3537 of file random.h.

References `std::numeric_limits<_Tp>::min()`.

5.640.4.3 operator()  

---

```
template<typename _UniformRandomNumberGenerator >
result_type std::bernoulli_distribution::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 3552 of file random.h.

## 5.640.4.4 p()

```
double std::bernoulli_distribution::p ( ) const [inline]
```

Returns the *p* parameter of the distribution.

Definition at line 3515 of file random.h.

Referenced by std::operator<<().

## 5.640.4.5 param() [1/2]

```
param_type std::bernoulli_distribution::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3522 of file random.h.

Referenced by std::operator>>().

## 5.640.4.6 param() [2/2]

```
void std::bernoulli_distribution::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3530 of file random.h.

#### 5.640.4.7 reset()

```
void std::bernoulli_distribution::reset ( ) [inline]
```

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3509 of file random.h.

### 5.640.5 Friends And Related Function Documentation

#### 5.640.5.1 operator==

```
bool operator== (
    const bernoulli\_distribution & __d1,
    const bernoulli\_distribution & __d2 ) [friend]
```

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3594 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.641 std::bernoulli\_distribution::param\_type Struct Reference

#### Public Types

- typedef [bernoulli\\_distribution](#) **distribution\_type**

#### Public Member Functions

- **param\_type** (double \_\_p=0.5)
- double **p** () const

#### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.641.1 Detailed Description

Parameter type.

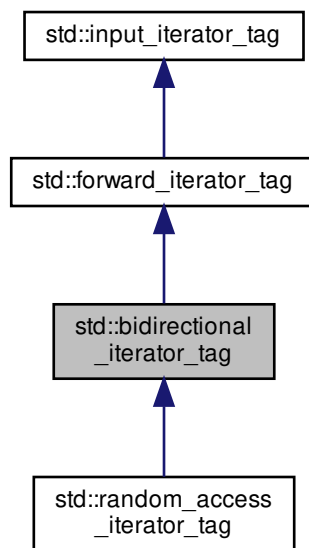
Definition at line 3459 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.642 std::bidirectional\_iterator\_tag Struct Reference

Inheritance diagram for std::bidirectional\_iterator\_tag:



## 5.642.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.

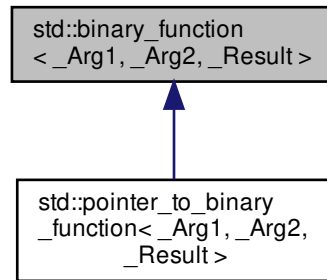
Definition at line 99 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 5.643 std::binary\_function< \_Arg1, \_Arg2, \_Result > Struct Template Reference

Inheritance diagram for std::binary\_function< \_Arg1, \_Arg2, \_Result >:



#### Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

#### 5.643.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 118 of file `stl_function.h`.

#### 5.643.2 Member Typedef Documentation

##### 5.643.2.1 first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary\_function< \_Arg1, \_Arg2, \_Result >::first\_argument\_type
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

## 5.643.2.2 result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 5.643.2.3 second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type
```

second\_argument\_type is the type of the second argument

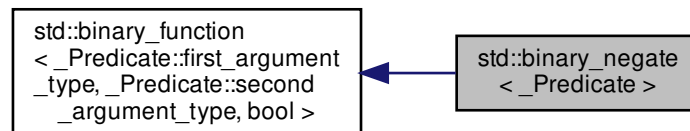
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.644 std::binary\_negate&lt; \_Predicate &gt; Class Template Reference

Inheritance diagram for std::binary\_negate< \_Predicate >:



## Public Types

- typedef \_Predicate::first\_argument\_type [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Predicate::second\_argument\_type [second\\_argument\\_type](#)

### Public Member Functions

- `_GLIBCXX14_CONSTEXPR binary_negate` (const `_Predicate` &\_\_x)
- `_GLIBCXX14_CONSTEXPR bool operator()` (const typename `_Predicate::first_argument_type` &\_\_x, const typename `_Predicate::second_argument_type` &\_\_y) const

### Protected Attributes

- `_Predicate _M_pred`

#### 5.644.1 Detailed Description

```
template<typename _Predicate>
class std::binary_negate<_Predicate >
```

One of the [negation functors](#).

Definition at line 1005 of file `stl_function.h`.

#### 5.644.2 Member Typedef Documentation

##### 5.644.2.1 first\_argument\_type

```
typedef _Predicate::first_argument_type std::binary\_function< _Predicate::first_argument_type , ↵
_Predicate::second_argument_type , bool >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 5.644.2.2 result\_type

```
typedef bool std::binary\_function< _Predicate::first_argument_type , _Predicate::second_argument_t↵
_type , bool >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

5.644.2.3 `second_argument_type`

```
typedef _Predicate::second_argument_type std::binary\_function< _Predicate::first_argument_type ,  
_Predicate::second_argument_type , bool >::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument

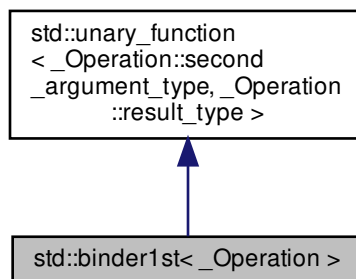
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

5.645 `std::binder1st<_Operation>` Class Template Reference

Inheritance diagram for `std::binder1st<_Operation>`:



## Public Types

- `typedef _Operation::second_argument_type` [argument\\_type](#)
- `typedef _Operation::result_type` [result\\_type](#)

## Public Member Functions

- **binder1st** (const `_Operation` &\_\_x, const typename `_Operation::first_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &\_\_x) const

## Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**



### 5.645.1 Detailed Description

```
template<typename _Operation>
class std::binder1st< _Operation >
```

One of the [binder functors](#).

Definition at line 108 of file binders.h.

### 5.645.2 Member Typedef Documentation

#### 5.645.2.1 argument\_type

```
typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type ,
_Operation::result_type >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

#### 5.645.2.2 result\_type

```
typedef _Operation::result_type std::unary_function< _Operation::second_argument_type , _Operation↵
::result_type >::result_type [inherited]
```

result\_type is the return type

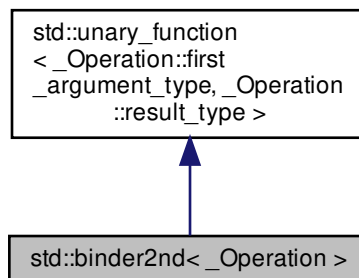
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [binders.h](#)

### 5.646 std::binder2nd< \_Operation > Class Template Reference

Inheritance diagram for std::binder2nd< \_Operation >:



### Public Types

- typedef \_Operation::first\_argument\_type [argument\\_type](#)
- typedef \_Operation::result\_type [result\\_type](#)

### Public Member Functions

- **binder2nd** (const \_Operation &\_\_x, const typename \_Operation::second\_argument\_type &\_\_y)
- \_Operation::result\_type **operator()** (const typename \_Operation::first\_argument\_type &\_\_x) const
- \_Operation::result\_type **operator()** (typename \_Operation::first\_argument\_type &\_\_x) const

### Protected Attributes

- \_Operation **op**
- \_Operation::second\_argument\_type **value**

#### 5.646.1 Detailed Description

```
template<typename _Operation>  
class std::binder2nd< _Operation >
```

One of the [binder functors](#).

Definition at line 143 of file binders.h.

#### 5.646.2 Member Typedef Documentation

##### 5.646.2.1 argument\_type

```
typedef _Operation::first_argument_type std::unary\_function< _Operation::first_argument_type , _↔  
Operation::result_type >::argument\_type [inherited]
```

[argument\\_type](#) is the type of the argument

Definition at line 108 of file stl\_function.h.

### 5.646.2.2 result\_type

```
typedef _Operation::result_type std::unary\_function< _Operation::first_argument_type , _Operation↵  
::result_type >::result\_type [inherited]
```

result\_type is the return type

Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [binders.h](#)

## 5.647 std::binomial\_distribution< \_IntType > Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_IntType [result\\_type](#)

### Public Member Functions

- **binomial\_distribution** (\_IntType \_\_t=\_IntType(1), double \_\_p=0.5)
- **binomial\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator , typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator , typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- \_IntType **t** () const

## Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::binomial_distribution< _IntType1 > &__x)`
- `bool operator== (const binomial_distribution &__d1, const binomial_distribution &__d2)`
- `template<typename _IntType1 , typename _CharT , typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::binomial_distribution< _IntType1 > &__x)`

## 5.647.1 Detailed Description

```
template<typename _IntType = int>
class std::binomial_distribution< _IntType >
```

A discrete binomial random number distribution.

The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3662 of file random.h.

## 5.647.2 Member Typedef Documentation

## 5.647.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 3665 of file random.h.

## 5.647.3 Member Function Documentation

## 5.647.3.1 max()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3777 of file random.h.

### 5.647.3.2 min()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3770 of file random.h.

### 5.647.3.3 operator() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::binomial_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 3785 of file random.h.

### 5.647.3.4 operator() [2/2]

```
template<typename _IntType >
template<typename _UniformRandomNumberGenerator >
binomial_distribution< _IntType >::result_type std::binomial_distribution< _IntType >::operator()
(
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param )
```

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise.  
NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1537 of file bits/random.tcc.

References `std::abs()`, `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::numeric_limits< _Tp >::max()`.

### 5.647.3.5 p()

```
template<typename _IntType = int>
double std::binomial_distribution< _IntType >::p ( ) const [inline]
```

Returns the distribution `p` parameter.

Definition at line 3748 of file random.h.

**5.647.3.6 param()** [1/2]

```
template<typename _IntType = int>
param_type std::binomial_distribution< _IntType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3755 of file random.h.

**5.647.3.7 param()** [2/2]

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3763 of file random.h.

**5.647.3.8 reset()**

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 3734 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

**5.647.3.9 t()**

```
template<typename _IntType = int>
_IntType std::binomial_distribution< _IntType >::t ( ) const [inline]
```

Returns the distribution `t` parameter.

Definition at line 3741 of file random.h.

## 5.647.4 Friends And Related Function Documentation

### 5.647.4.1 operator<<

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::binomial_distribution< _IntType1 > & __x ) [friend]
```

Inserts a binomial\_distribution random number distribution \_\_x into the output stream \_\_os.

#### Parameters

__os	An output stream.
__x	A binomial_distribution random number distribution.

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

### 5.647.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
    const binomial_distribution< _IntType > & __d1,
    const binomial_distribution< _IntType > & __d2 ) [friend]
```

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3821 of file random.h.

### 5.647.4.3 operator>>

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::binomial_distribution< _IntType1 > & __x ) [friend]
```

Extracts a binomial\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>binomial_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.648 `std::binomial_distribution< _IntType >::param_type` Struct Reference

## Public Types

- typedef `binomial_distribution< _IntType >` **distribution\_type**

## Public Member Functions

- **param\_type** (`_IntType __t=_IntType(1), double __p=0.5`)
- double **p** () const
- `_IntType t` () const

## Friends

- class **binomial\_distribution< \_IntType >**
- bool **operator!=** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 5.648.1 Detailed Description

```
template<typename _IntType = int>
struct std::binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3672 of file `random.h`.

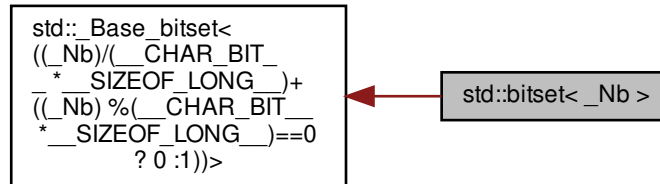
The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)



## 5.649 std::bitset<\_Nb> Class Template Reference

Inheritance diagram for std::bitset<\_Nb>:



### Classes

- class [reference](#)

### Public Member Functions

- constexpr [bitset](#) () noexcept
- constexpr [bitset](#) (unsigned long long \_\_val) noexcept
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position=0)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT >  
[bitset](#) (const \_CharT \* \_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- size\_t [\\_Find\\_first](#) () const noexcept
- size\_t [\\_Find\\_next](#) (size\_t \_\_prev) const noexcept
- template<class \_CharT, class \_Traits >  
void [\\_M\\_copy\\_from\\_ptr](#) (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT, \_CharT)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_from\\_string](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_from\\_string](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_pos, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &, \_CharT, \_CharT) const
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s) const
- bool [all](#) () const noexcept

- `bool any ()` const noexcept
  - `size_t count ()` const noexcept
  - `bitset<_Nb> & flip ()` noexcept
  - `bitset<_Nb> & flip (size_t __position)`
  - `bool none ()` const noexcept
  - `bitset<_Nb> operator~ ()` const noexcept
  - `bitset<_Nb> & reset ()` noexcept
  - `bitset<_Nb> & reset (size_t __position)`
  - `bitset<_Nb> & set ()` noexcept
  - `bitset<_Nb> & set (size_t __position, bool __val=true)`
  - `constexpr size_t size ()` const noexcept
  - `bool test (size_t __position)` const
  - `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_string<_CharT, _Traits, _Alloc> to_string ()` const
  - `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_string<_CharT, _Traits, _Alloc> to_string (_CharT __zero, _CharT __one=_CharT('1'))` const
  - `template<class _CharT, class _Traits>`  
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string ()` const
  - `template<class _CharT, class _Traits>`  
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string (_CharT __zero, _CharT __one=_CharT('1'))` const
  - `template<class _CharT>`  
`std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string ()` const
  - `template<class _CharT>`  
`std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string (_CharT __zero, _CharT __one=_CharT('1'))` const
  - `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string ()` const
  - `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string (char __zero, char __one='1')` const
  - `unsigned long long to_ulong ()` const
  - `unsigned long to_ulong ()` const
- 
- `bitset<_Nb> & operator&= (const bitset<_Nb> &__rhs)` noexcept
  - `bitset<_Nb> & operator|= (const bitset<_Nb> &__rhs)` noexcept
  - `bitset<_Nb> & operator^= (const bitset<_Nb> &__rhs)` noexcept
- 
- `bitset<_Nb> & operator<<= (size_t __position)` noexcept
  - `bitset<_Nb> & operator>>= (size_t __position)` noexcept
- 
- `bitset<_Nb> & _Unchecked_set (size_t __pos)` noexcept
  - `bitset<_Nb> & _Unchecked_set (size_t __pos, int __val)` noexcept
  - `bitset<_Nb> & _Unchecked_reset (size_t __pos)` noexcept

- `bitset<_Nb> &_Unchecked_flip` (size\_t \_\_pos) noexcept
- `constexpr bool _Unchecked_test` (size\_t \_\_pos) const noexcept

- `reference operator[]` (size\_t \_\_position)
- `constexpr bool operator[]` (size\_t \_\_position) const

- `bool operator==` (const `bitset<_Nb>` &\_\_rhs) const noexcept
- `bool operator!=` (const `bitset<_Nb>` &\_\_rhs) const noexcept

- `bitset<_Nb> operator<<` (size\_t \_\_position) const noexcept
- `bitset<_Nb> operator>>` (size\_t \_\_position) const noexcept

#### Private Member Functions

- `bool _M_are_all` () const noexcept
- `void _M_do_and` (const `_Base_bitset<_Nw>` &\_\_x) noexcept
- `size_t _M_do_count` () const noexcept
- `size_t _M_do_find_first` (size\_t) const noexcept
- `size_t _M_do_find_next` (size\_t, size\_t) const noexcept
- `void _M_do_flip` () noexcept
- `void _M_do_left_shift` (size\_t \_\_shift) noexcept
- `void _M_do_or` (const `_Base_bitset<_Nw>` &\_\_x) noexcept
- `void _M_do_reset` () noexcept
- `void _M_do_right_shift` (size\_t \_\_shift) noexcept
- `void _M_do_set` () noexcept
- `unsigned long long _M_do_to_ullong` () const
- `unsigned long _M_do_to_ulong` () const
- `void _M_do_xor` (const `_Base_bitset<_Nw>` &\_\_x) noexcept
- `const _WordT * _M_getdata` () const noexcept
- `_WordT & _M_getword` (size\_t \_\_pos) noexcept
- `constexpr _WordT _M_getword` (size\_t \_\_pos) const noexcept
- `_WordT & _M_hiword` () noexcept
- `constexpr _WordT _M_hiword` () const noexcept
- `bool _M_is_any` () const noexcept
- `bool _M_is_equal` (const `_Base_bitset<_Nw>` &\_\_x) const noexcept

#### Static Private Member Functions

- `static constexpr _WordT _S_maskbit` (size\_t \_\_pos) noexcept
- `static constexpr size_t _S_whichbit` (size\_t \_\_pos) noexcept
- `static constexpr size_t _S_whichbyte` (size\_t \_\_pos) noexcept
- `static constexpr size_t _S_whichword` (size\_t \_\_pos) noexcept

## Private Attributes

- `_WordT` `_M_w` [`_Nw`]

## Friends

- class **reference**
- struct **std::hash**< **bitset** >

## 5.649.1 Detailed Description

```
template<size_t _Nb>
class std::bitset<_Nb>
```

The `bitset` class represents a *fixed-size* sequence of bits.

(Note that `bitset` does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let *B* be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage. *B* - *Nb* bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as *a simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant* / *right-hand* position, and the bit at index *Nb*-1 in the *most significant* / *left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
    long    a = 'a';
    bitset<10> b(a);
    cout << "b('a') is " << b << endl;
    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_containers.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_containers.html) for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the `bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` *is* a regular array, and is indexed as such. This is carefully encapsulated.

Definition at line 751 of file `bitset`.

## 5.649.2 Constructor & Destructor Documentation

### 5.649.2.1 `bitset()` [1/5]

```
template<size_t _Nb>
constexpr std::bitset< _Nb >::bitset ( ) [inline], [noexcept]
```

All bits set to zero.

Definition at line 865 of file `bitset`.

### 5.649.2.2 `bitset()` [2/5]

```
template<size_t _Nb>
constexpr std::bitset< _Nb >::bitset (
    unsigned long long __val ) [inline], [noexcept]
```

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 870 of file `bitset`.

### 5.649.2.3 `bitset()` [3/5]

```
template<size_t _Nb>
template<class _CharT , class _Traits , class _Alloc >
std::bitset< _Nb >::bitset (
    const std::basic_string< _CharT, _Traits, _Alloc > & __s,
    size_t __position = 0 ) [inline], [explicit]
```

Use a subset of a string.

#### Parameters

<code>__s</code>	A string of <i>0</i> and <i>1</i> characters.
<code>__position</code>	Index of the first character in <code>__s</code> to use; defaults to zero.

#### Exceptions

<code>std::out_of_range</code>	If <i>pos</i> is bigger the size of <code>__s</code> .
<code>std::invalid_argument</code>	If a character appears in the string which is neither <i>0</i> nor <i>1</i> .

Definition at line 889 of file `bitset`.

## 5.649.2.4 bitset() [4/5]

```
template<size_t _Nb>
template<class _CharT , class _Traits , class _Alloc >
std::bitset< _Nb >::bitset (
    const std::basic_string< _CharT, _Traits, _Alloc > & __s,
    size_t __position,
    size_t __n ) [inline]
```

Use a subset of a string.

## Parameters

<code>__s</code>	A string of 0 and 1 characters.
<code>__position</code>	Index of the first character in <code>__s</code> to use.
<code>__n</code>	The number of characters to copy.

## Exceptions

<code>std::out_of_range</code>	If <code>__position</code> is bigger the size of <code>__s</code> .
<code>std::invalid_argument</code>	If a character appears in the string which is neither 0 nor 1.

Definition at line 910 of file `bitset`.

## 5.649.2.5 bitset() [5/5]

```
template<size_t _Nb>
template<typename _CharT >
std::bitset< _Nb >::bitset (
    const _CharT * __str,
    typename std::basic_string< _CharT >::size_type __n = std::basic_string<_CharT>::npos,
    _CharT __zero = _CharT('0'),
    _CharT __one = _CharT('1') ) [inline], [explicit]
```

Construct from a character array.

## Parameters

<code>__str</code>	An array of characters <i>zero</i> and <i>one</i> .
<code>__n</code>	The number of characters to use.
<code>__zero</code>	The character corresponding to the value 0.
<code>__one</code>	The character corresponding to the value 1.

**Exceptions**

<code>std::invalid_argument</code>	If a character appears in the string which is neither <code>__zero</code> nor <code>__one</code> .
------------------------------------	--

Definition at line 942 of file `bitset`.

**5.649.3 Member Function Documentation****5.649.3.1 all()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::all ( ) const [inline], [noexcept]
```

Tests whether all the bits are on.

**Returns**

True if all the bits are set.

Definition at line 1330 of file `bitset`.

**5.649.3.2 any()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::any ( ) const [inline], [noexcept]
```

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

Definition at line 1338 of file `bitset`.

**5.649.3.3 count()**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::count ( ) const [inline], [noexcept]
```

Returns the number of bits which are set.

Definition at line 1291 of file `bitset`.

#### 5.649.3.4 flip() [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::flip ( ) [inline], [noexcept]
```

Toggles every bit to its opposite value.

Definition at line 1119 of file `bitset`.

#### 5.649.3.5 flip() [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::flip (
    size_t __position ) [inline]
```

Toggles a given bit to its opposite value.

##### Parameters

<code>__position</code>	The index of the bit.
-------------------------	-----------------------

##### Exceptions

<code>std::out_of_range</code>	If <i>pos</i> is bigger the size of the set.
--------------------------------	--

Definition at line 1132 of file `bitset`.

#### 5.649.3.6 none()

```
template<size_t _Nb>
bool std::bitset<_Nb>::none ( ) const [inline], [noexcept]
```

Tests whether any of the bits are on.

##### Returns

True if none of the bits are set.

Definition at line 1346 of file `bitset`.



#### 5.649.3.7 operator!=()

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator!= (
    const bitset< _Nb > & __rhs ) const [inline], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1306 of file `bitset`.

#### 5.649.3.8 operator&=()

```
template<size_t _Nb>
bitset<_Nb>& std::bitset< _Nb >::operator&= (
    const bitset< _Nb > & __rhs ) [inline], [noexcept]
```

Operations on bitsets.

##### Parameters

<code>__rhs</code>	A same-sized <code>bitset</code> .
--------------------	------------------------------------

These should be self-explanatory.

Definition at line 968 of file `bitset`.

#### 5.649.3.9 operator<<()

```
template<size_t _Nb>
bitset<_Nb> std::bitset< _Nb >::operator<< (
    size_t __position ) const [inline], [noexcept]
```

Self-explanatory.

Definition at line 1352 of file `bitset`.

#### 5.649.3.10 operator<<=()

```
template<size_t _Nb>
bitset<_Nb>& std::bitset< _Nb >::operator<<= (
    size_t __position ) [inline], [noexcept]
```

Operations on bitsets.

## Parameters

<code>__position</code>	The number of places to shift.
-------------------------	--------------------------------

These should be self-explanatory.

Definition at line 997 of file `bitset`.

## 5.649.3.11 operator==( )

```
template<size_t _Nb>
bool std::bitset<_Nb>::operator==(
    const bitset<_Nb> & __rhs ) const [inline], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1302 of file `bitset`.

## 5.649.3.12 operator&gt;&gt;( )

```
template<size_t _Nb>
bitset<_Nb> std::bitset<_Nb>::operator>> (
    size_t __position ) const [inline], [noexcept]
```

Self-explanatory.

Definition at line 1356 of file `bitset`.

## 5.649.3.13 operator&gt;&gt;=( )

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator>>= (
    size_t __position ) [inline], [noexcept]
```

Operations on bitsets.

## Parameters

<code>__position</code>	The number of places to shift.
-------------------------	--------------------------------

These should be self-explanatory.

Definition at line 1010 of file `bitset`.

**5.649.3.14** `operator[]()` [1/2]

```
template<size_t _Nb>
reference std::bitset<_Nb>::operator[] (
    size_t __position ) [inline]
```

Array-indexing support.

**Parameters**

<code>__position</code>	Index into the bitset.
-------------------------	------------------------

**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1159 of file `bitset`.

**5.649.3.15** `operator[]()` [2/2]

```
template<size_t _Nb>
constexpr bool std::bitset<_Nb>::operator[] (
    size_t __position ) const [inline]
```

Array-indexing support.

**Parameters**

<code>__position</code>	Index into the bitset.
-------------------------	------------------------

**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1163 of file `bitset`.

**5.649.3.16 operator^=()**

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator^= (
    const bitset<_Nb> & __rhs ) [inline], [noexcept]
```

Operations on bitsets.

**Parameters**

<code>__rhs</code>	A same-sized bitset.
--------------------	----------------------

These should be self-explanatory.

Definition at line 982 of file `bitset`.

**5.649.3.17 operator" |=()**

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator|= (
    const bitset<_Nb> & __rhs ) [inline], [noexcept]
```

Operations on bitsets.

**Parameters**

<code>__rhs</code>	A same-sized bitset.
--------------------	----------------------

These should be self-explanatory.

Definition at line 975 of file `bitset`.

**5.649.3.18 operator~()**

```
template<size_t _Nb>
bitset<_Nb> std::bitset< _Nb >::operator~ ( ) const [inline], [noexcept]
```

See the no-argument flip().

Definition at line 1140 of file `bitset`.

**5.649.3.19 reset()** [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset< _Nb >::reset ( ) [inline], [noexcept]
```

Sets every bit to false.

Definition at line 1095 of file `bitset`.

**5.649.3.20 reset()** [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset< _Nb >::reset (
    size_t __position ) [inline]
```

Sets a given bit to false.

**Parameters**

<code>__position</code>	The index of the bit.
-------------------------	-----------------------

**Exceptions**

<code>std::out_of_range</code>	If <code>pos</code> is bigger the size of the set.
--------------------------------	--

Same as writing `set (pos, false)`.

Definition at line 1109 of file `bitset`.

**5.649.3.21 set()** [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset< _Nb >::set ( ) [inline], [noexcept]
```

Sets every bit to true.

Definition at line 1071 of file `bitset`.

#### 5.649.3.22 `set()` [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::set (
    size_t __position,
    bool __val = true ) [inline]
```

Sets a given bit to a particular value.

##### Parameters

<code>__position</code>	The index of the bit.
<code>__val</code>	Either true or false, defaults to true.

##### Exceptions

<code>std::out_of_range</code>	If <i>pos</i> is bigger the size of the set.
--------------------------------	--

Definition at line 1085 of file `bitset`.

#### 5.649.3.23 `size()`

```
template<size_t _Nb>
constexpr size_t std::bitset<_Nb>::size ( ) const [inline], [noexcept]
```

Returns the total number of bits.

Definition at line 1296 of file `bitset`.

#### 5.649.3.24 `test()`

```
template<size_t _Nb>
bool std::bitset<_Nb>::test (
    size_t __position ) const [inline]
```

Tests the value of a bit.

**Parameters**

<code>__position</code>	The index of a bit.
-------------------------	---------------------

**Returns**

The value at *pos*.

**Exceptions**

<code>std::out_of_range</code>	If <i>pos</i> is bigger the size of the set.
--------------------------------	--

Definition at line 1317 of file `bitset`.

**5.649.3.25 to\_string()**

```
template<size_t _Nb>
template<class _CharT , class _Traits , class _Alloc >
std::basic_string<_CharT, _Traits, _Alloc> std::bitset< _Nb >::to_string ( ) const [inline]
```

Returns a character interpretation of the `bitset`.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1193 of file `bitset`.

**5.649.3.26 to\_ulong()**

```
template<size_t _Nb>
unsigned long std::bitset< _Nb >::to_ulong ( ) const [inline]
```

Returns a numerical interpretation of the `bitset`.

**Returns**

The integral equivalent of the bits.

## Exceptions

<code>std::overflow_error</code>	If there are too many bits to be represented in an unsigned long.
----------------------------------	---

Definition at line 1174 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

5.650 `std::bitset<_Nb>::reference` Class Reference

## Public Member Functions

- **reference** ([bitset](#) &\_\_b, `size_t` \_\_pos) noexcept
- [reference](#) & **flip** () noexcept
- **operator bool** () const noexcept
- [reference](#) & **operator=** (`bool` \_\_x) noexcept
- [reference](#) & **operator=** (const [reference](#) &\_\_j) noexcept
- `bool` **operator~** () const noexcept

## Friends

- class **bitset**

## 5.650.1 Detailed Description

```
template<size_t _Nb>
class std::bitset<_Nb>::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

Definition at line 802 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)



## 5.651 std::cauchy\_distribution<\_RealType> Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **cauchy\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **cauchy\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- bool **operator==** (const [cauchy\\_distribution](#) &\_\_d1, const [cauchy\\_distribution](#) &\_\_d2)

#### 5.651.1 Detailed Description

```
template<typename _RealType = double>
class std::cauchy_distribution<_RealType>
```

A cauchy\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|a,b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2794 of file random.h.

### 5.651.2 Member Typedef Documentation

#### 5.651.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::cauchy_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2797 of file random.h.

### 5.651.3 Member Function Documentation

#### 5.651.3.1 max()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2890 of file random.h.

References std::numeric\_limits< \_Tp >::max().

#### 5.651.3.2 min()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2883 of file random.h.

References std::numeric\_limits< \_Tp >::lowest().

#### 5.651.3.3 operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::cauchy_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 2898 of file random.h.

#### 5.651.3.4 param() [1/2]

```
template<typename _RealType = double>
param_type std::cauchy_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2868 of file random.h.

Referenced by std::operator>>().

#### 5.651.3.5 param() [2/2]

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2876 of file random.h.

#### 5.651.3.6 reset()

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 2850 of file random.h.

### 5.651.4 Friends And Related Function Documentation

#### 5.651.4.1 operator==

```
template<typename _RealType = double>
bool operator== (
    const cauchy_distribution< _RealType > & __d1,
    const cauchy_distribution< _RealType > & __d2 ) [friend]
```

Return true if two Cauchy distributions have the same parameters.

Definition at line 2933 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.652 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `cauchy_distribution<_RealType>` `distribution_type`

## Public Member Functions

- `param_type` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

## Friends

- bool `operator!=` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 5.652.1 Detailed Description

```
template<typename _RealType = double>
struct std::cauchy_distribution<_RealType>::param_type
```

Parameter type.

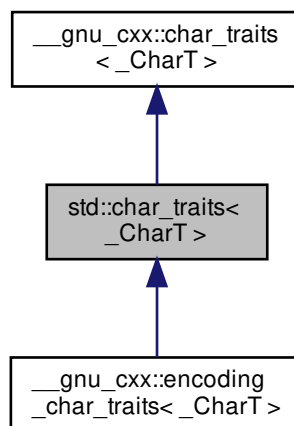
Definition at line 2804 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.653 `std::char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `std::char_traits<_CharT>`:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types< _CharT >::int_type` **int\_type**
- typedef `_Char_types< _CharT >::off_type` **off\_type**
- typedef `_Char_types< _CharT >::pos_type` **pos\_type**
- typedef `_Char_types< _CharT >::state_type` **state\_type**

## Static Public Member Functions

- static `_GLIBCXX14_CONSTEXPR` void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static `_GLIBCXX14_CONSTEXPR` int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `constexpr int_type` **eof** ()
- static `constexpr bool` **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static `constexpr bool` **eq\_int\_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `_GLIBCXX14_CONSTEXPR` `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)
- static `_GLIBCXX14_CONSTEXPR` `std::size_t` **length** (`const char_type *__s`)
- static `constexpr bool` **lt** (`const char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `constexpr int_type` **not\_eof** (`const int_type &__c`)
- static `constexpr char_type` **to\_char\_type** (`const int_type &__c`)
- static `constexpr int_type` **to\_int\_type** (`const char_type &__c`)

### 5.653.1 Detailed Description

```
template<class _CharT>
struct std::char_traits< _CharT >
```

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 271 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

5.654 `std::char_traits<__gnu_cxx::character<_Value, _Int, _St>>` Struct Template Reference

## Public Types

- typedef `__gnu_cxx::character<_Value, _Int, _St>` **char\_type**
- typedef `char_type::int_type` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `fpos<state_type>` **pos\_type**
- typedef `char_type::state_type` **state\_type**

## Static Public Member Functions

- static void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **assign** (`char_type` \* \_\_s, size\_t \_\_n, `char_type` \_\_a)
- static int **compare** (const `char_type` \* \_\_s1, const `char_type` \* \_\_s2, size\_t \_\_n)
- static `char_type` \* **copy** (`char_type` \* \_\_s1, const `char_type` \* \_\_s2, size\_t \_\_n)
- static int\_type **eof** ()
- static bool **eq** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const `char_type` \* **find** (const `char_type` \* \_\_s, size\_t \_\_n, const `char_type` &\_\_a)
- static size\_t **length** (const `char_type` \* \_\_s)
- static bool **lt** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **move** (`char_type` \* \_\_s1, const `char_type` \* \_\_s2, size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static `char_type` **to\_char\_type** (const int\_type &\_\_i)
- static int\_type **to\_int\_type** (const `char_type` &\_\_c)

## 5.654.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St>
struct std::char_traits<__gnu_cxx::character<_Value, _Int, _St>>
```

`char_traits<__gnu_cxx::character>` specialization.

Definition at line 97 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

5.655 `std::char_traits<char>` Struct Template Reference

## Public Types

- typedef `char` **char\_type**
- typedef `int` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `streampos` **pos\_type**
- typedef `mbstate_t` **state\_type**

### Static Public Member Functions

- static \_GLIBCXX17\_CONSTEXPR void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **assign** (char\_type \* \_\_s, size\_t \_\_n, char\_type \_\_a)
- static \_GLIBCXX17\_CONSTEXPR int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static \_GLIBCXX17\_CONSTEXPR const char\_type \* **find** (const char\_type \* \_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static \_GLIBCXX17\_CONSTEXPR size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

#### 5.655.1 Detailed Description

```
template<>
struct std::char_traits< char >
```

##### 21.1.3.1 char\_traits specializations

Definition at line 277 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

#### 5.656 std::char\_traits< wchar\_t > Struct Template Reference

##### Public Types

- typedef wchar\_t **char\_type**
- typedef wint\_t **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [wstreampos](#) **pos\_type**
- typedef mbstate\_t **state\_type**

## Static Public Member Functions

- static \_GLIBCXX17\_CONSTEXPR void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static \_GLIBCXX17\_CONSTEXPR int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static \_GLIBCXX17\_CONSTEXPR const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static \_GLIBCXX17\_CONSTEXPR size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

## 5.656.1 Detailed Description

```
template<>
struct std::char_traits<wchar_t>
```

## 21.1.3.2 char\_traits specializations

Definition at line 397 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.657 std::chi\_squared\_distribution&lt;\_RealType&gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)



## Public Member Functions

- **chi\_squared\_distribution** (\_RealType \_\_n=\_RealType(1))
- **chi\_squared\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- \_RealType n () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) param () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)
- bool [operator==](#) (const [chi\\_squared\\_distribution](#) &\_\_d1, const [chi\\_squared\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)

## 5.657.1 Detailed Description

```
template<typename _RealType = double>
class std::chi_squared_distribution< _RealType >
```

A [chi\\_squared\\_distribution](#) random number distribution.

The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Definition at line 2574 of file random.h.

## 5.657.2 Member Typedef Documentation

### 5.657.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::chi_squared_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2577 of file random.h.

## 5.657.3 Member Function Documentation

### 5.657.3.1 max()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2664 of file random.h.

References std::numeric\_limits<\_Tp>::max().

### 5.657.3.2 min()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2657 of file random.h.

### 5.657.3.3 operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::chi_squared_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 2672 of file random.h.

#### 5.657.3.4 param() [1/2]

```
template<typename _RealType = double>
param_type std::chi_squared_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2637 of file random.h.

#### 5.657.3.5 param() [2/2]

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2645 of file random.h.

References `std::gamma_distribution< _RealType >::param()`.

#### 5.657.3.6 reset()

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 2623 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

### 5.657.4 Friends And Related Function Documentation

#### 5.657.4.1 operator<<

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::chi_squared_distribution< _RealType1 > & __x ) [friend]
```

Inserts a `chi_squared_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A chi_squared_distribution random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

## 5.657.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
    const chi_squared_distribution<_RealType> & __d1,
    const chi_squared_distribution<_RealType> & __d2 ) [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2723 of file random.h.

## 5.657.4.3 operator&gt;&gt;

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::chi_squared_distribution<_RealType1> & __x ) [friend]
```

Extracts a chi\_squared\_distribution random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A chi_squared_distribution random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.658 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `chi_squared_distribution<_RealType>` **distribution\_type**

### Public Member Functions

- **param\_type** (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

### Friends

- bool **operator!=** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.658.1 Detailed Description

```
template<typename _RealType = double>
struct std::chi_squared_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2584 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.659 `std::chrono::_V2::steady_clock` Struct Reference

### Public Types

- typedef `chrono::nanoseconds` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**
- typedef `chrono::time_point<steady_clock, duration>` **time\_point**

### Static Public Member Functions

- static `time_point` **now** () noexcept

#### Static Public Attributes

- static constexpr bool **is\_steady**

#### 5.659.1 Detailed Description

Monotonic clock.

Time returned has the property of only increasing at a uniform rate.

Definition at line 857 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.660 std::chrono::\_V2::system\_clock Struct Reference

#### Public Types

- typedef [chrono::nanoseconds](#) **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef [chrono::time\\_point](#)< [system\\_clock](#), [duration](#) > **time\_point**

#### Static Public Member Functions

- static [time\\_point](#) **from\_time\_t** (std::time\_t \_\_t) noexcept
- static [time\\_point](#) **now** () noexcept
- static std::time\_t **to\_time\_t** (const [time\\_point](#) &\_\_t) noexcept

#### Static Public Attributes

- static constexpr bool **is\_steady**

#### 5.660.1 Detailed Description

System clock.

Time returned represents wall time from the system-wide clock.

Definition at line 818 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.661 std::chrono::duration<\_Rep, \_Period> Struct Template Reference

### Public Types

- typedef `_Period` **period**
- typedef `_Rep` **rep**

### Public Member Functions

- **duration** (const `duration` &)=default
- template<typename `_Rep2` , typename = `_Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __not_<__is_<float<_Rep2>>>>>`  
constexpr **duration** (const `_Rep2` &\_\_rep)
- template<typename `_Rep2` , typename `_Period2` , typename = `_Require< __or_<__is_float<rep>, __and_<__is_harmonic<_Period2>, __not_<__is_float<_Rep2>>>>>`  
constexpr **duration** (const `duration`< `_Rep2`, `_Period2` > &\_\_d)
- constexpr `rep` **count** () const
- template<typename `_Rep2` = `rep`>  
`_GLIBCXX17_CONSTEXPR` `enable_if<!treat_as_floating_point< _Rep2 >::value, duration & >::type` **operator%=** (const `rep` &\_\_rhs)
- template<typename `_Rep2` = `rep`>  
`_GLIBCXX17_CONSTEXPR` `enable_if<!treat_as_floating_point< _Rep2 >::value, duration & >::type` **operator%=** (const `duration` &\_\_d)
- `_GLIBCXX17_CONSTEXPR` `duration` & **operator\*=** (const `rep` &\_\_rhs)
- constexpr `duration` **operator+** () const
- `_GLIBCXX17_CONSTEXPR` `duration` & **operator++** ()
- `_GLIBCXX17_CONSTEXPR` `duration` **operator++** (int)
- `_GLIBCXX17_CONSTEXPR` `duration` & **operator+=** (const `duration` &\_\_d)
- constexpr `duration` **operator-** () const
- `_GLIBCXX17_CONSTEXPR` `duration` & **operator--** ()
- `_GLIBCXX17_CONSTEXPR` `duration` **operator--** (int)
- `_GLIBCXX17_CONSTEXPR` `duration` & **operator-=** (const `duration` &\_\_d)
- `_GLIBCXX17_CONSTEXPR` `duration` & **operator/=** (const `rep` &\_\_rhs)
- `duration` & **operator=** (const `duration` &)=default

### Static Public Member Functions

- static constexpr `duration` **max** () noexcept
- static constexpr `duration` **min** () noexcept
- static constexpr `duration` **zero** () noexcept

#### 5.661.1 Detailed Description

```
template<typename _Rep, typename _Period>
struct std::chrono::duration< _Rep, _Period >
```

`duration`

Definition at line 64 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

## 5.662 std::chrono::duration\_values< \_Rep > Struct Template Reference

### Static Public Member Functions

- static constexpr \_Rep **max** () noexcept
- static constexpr \_Rep **min** () noexcept
- static constexpr \_Rep **zero** () noexcept

### 5.662.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::duration_values< _Rep >
```

duration\_values

Definition at line 275 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.663 std::chrono::time\_point< \_Clock, \_Dur > Struct Template Reference

### Public Types

- typedef \_Clock **clock**
- typedef \_Dur **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**

### Public Member Functions

- constexpr **time\_point** (const duration &\_\_dur)
- template<typename \_Dur2, typename = \_Require<is\_convertible<\_Dur2, \_Dur>>>> constexpr **time\_point** (const [time\\_point](#)< clock, \_Dur2 > &\_\_t)
- \_GLIBCXX17\_CONSTEXPR [time\\_point](#) & **operator+=** (const duration &\_\_dur)
- \_GLIBCXX17\_CONSTEXPR [time\\_point](#) & **operator-=** (const duration &\_\_dur)
- constexpr duration **time\_since\_epoch** () const

### Static Public Member Functions

- static constexpr [time\\_point](#) **max** () noexcept
- static constexpr [time\\_point](#) **min** () noexcept



### 5.663.1 Detailed Description

```
template<typename _Clock, typename _Dur>
struct std::chrono::time_point< _Clock, _Dur >
```

time\_point

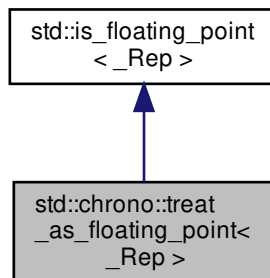
Definition at line 67 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

### 5.664 std::chrono::treat\_as\_floating\_point< \_Rep > Struct Template Reference

Inheritance diagram for std::chrono::treat\_as\_floating\_point< \_Rep >:



### 5.664.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::treat_as_floating_point< _Rep >
```

treat\_as\_floating\_point

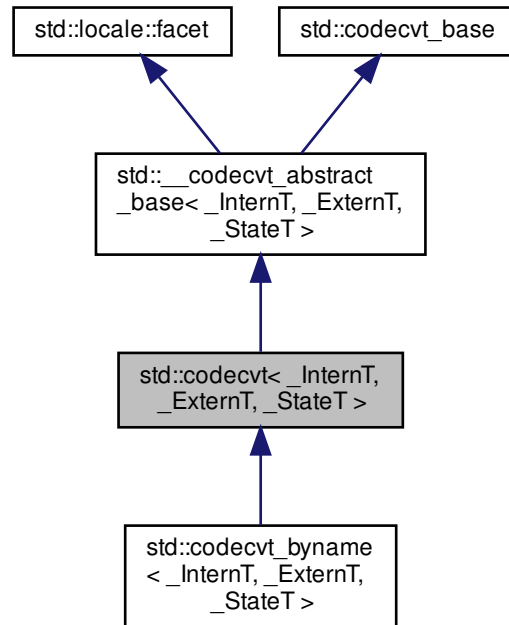
Definition at line 207 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.665 std::codecvt&lt; \_InternT, \_ExternT, \_StateT &gt; Class Template Reference

Inheritance diagram for std::codecvt< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

### Static Public Attributes

- static [locale::id](#) **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

#### 5.665.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt< _InternT, _ExternT, _StateT >
```

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 274 of file codecvt.h.

#### 5.665.2 Member Function Documentation

5.665.2.1 `do_out()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<\_InternT, \_ExternT, \_StateT >::do\_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

## See also

`out` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

5.665.2.2 `in()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

**5.665.2.3 out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

## 5.665.2.4 unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

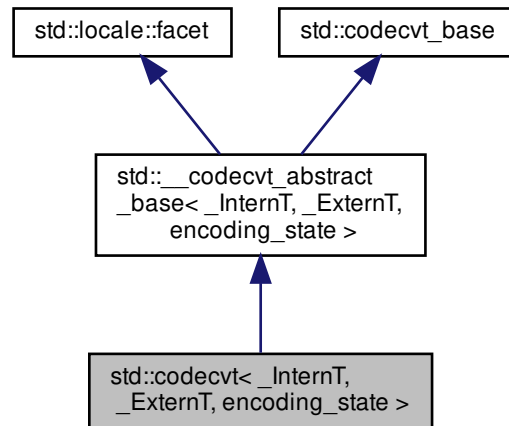
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.666 std::codecvt<\_InternT, \_ExternT, encoding\_state > Class Template Reference

Inheritance diagram for std::codecvt<\_InternT, \_ExternT, encoding\_state >:

**Public Types**

- typedef state\_type::descriptor\_type **descriptor\_type**
- typedef \_ExternT **extern\_type**
- typedef \_InternT **intern\_type**
- typedef codecvt\_base::result **result**
- typedef [\\_\\_gnu\\_cxx::encoding\\_state](#) **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (state\_type &\_\_enc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static locale::id **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.666.1 Detailed Description

```
template<typename _InternT, typename _ExternT>
class std::codecvt<_InternT, _ExternT, encoding_state >
```

codecvt<InternT, \_ExternT, encoding\_state> specialization.

Definition at line 233 of file codecvt\_specializations.h.



## 5.666.2 Member Function Documentation

### 5.666.2.1 do\_out()

```
template<typename _InternT, typename _ExternT >
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type * & __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type * & __to_next ) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`do_in` for more information.

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >`.

Definition at line 309 of file `codecvt_specializations.h`.

### 5.666.2.2 in()

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type * & __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type * & __to_next ) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

## 5.666.2.3 out()

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**5.666.2.4 unshift()**

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

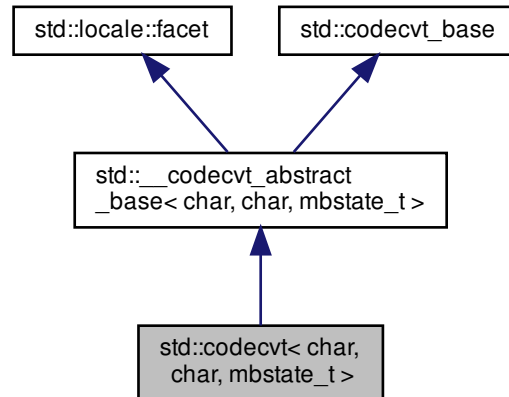
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.667 std::codecvt&lt; char, char, mbstate\_t &gt; Class Template Reference

Inheritance diagram for std::codecvt< char, char, mbstate\_t >:



## Public Types

- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

### Friends

- class **messages**< char >

#### 5.667.1 Detailed Description

```
template<>
class std::codecvt< char, char, mbstate_t >
```

class codecvt<char, char, mbstate\_t> specialization.

Definition at line 338 of file codecvt.h.

#### 5.667.2 Member Function Documentation

5.667.2.1 `do_out()`

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const    [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

**See also**

`out` for more information.

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

5.667.2.2 `in()`

```
result std::__codecvt_abstract_base< char , char , mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const    [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

5.667.2.3 `out()`

```
result std::__codecvt_abstract_base< char , char , mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**5.667.2.4 unshift()**

```
result std::__codecvt_abstract_base< char , char , mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

Definition at line 155 of file codecvt.h.

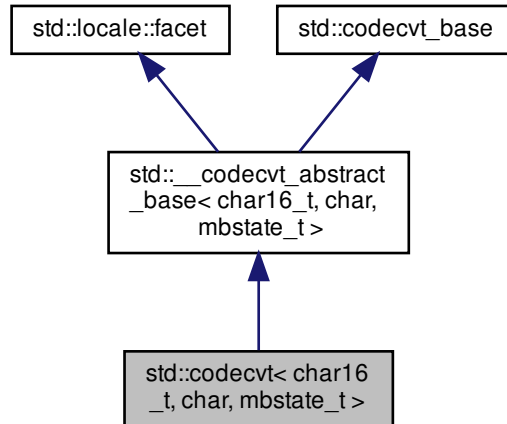
The documentation for this class was generated from the following file:

- [codecvt.h](#)



## 5.668 std::codecvt< char16\_t, char, mbstate\_t > Class Template Reference

Inheritance diagram for std::codecvt< char16\_t, char, mbstate\_t >:



### Public Types

- typedef char **extern\_type**
- typedef char16\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static **locale::id** id

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.668.1 Detailed Description

```
template<>
class std::codecvt< char16_t, char, mbstate_t >
```

Class codecvt<char16\_t, char, mbstate\_t> specialization.

Converts between UTF-16 and UTF-8.

Definition at line 468 of file codecvt.h.

## 5.668.2 Member Function Documentation

### 5.668.2.1 do\_out()

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const    [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char16\_t, char, mbstate\_t >.

### 5.668.2.2 in()

```
result std::\_\_codecvt\_abstract\_base< char16_t , char , mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const    [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling [codecvt::do\\_in](#).

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of [codecvt\\_base::result](#). If all the input is converted, returns [codecvt\\_base::ok](#). If no conversion is necessary, returns [codecvt\\_base::noconv](#). If the input ends early or there is insufficient space in the output, returns [codecvt\\_base::partial](#). Otherwise the conversion failed and [codecvt\\_base::error](#) is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

## 5.668.2.3 out()

```
result std::__codecvt_abstract_base< char16_t , char , mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**5.668.2.4 unshift()**

```
result std::__codecvt_abstract_base< char16_t , char , mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

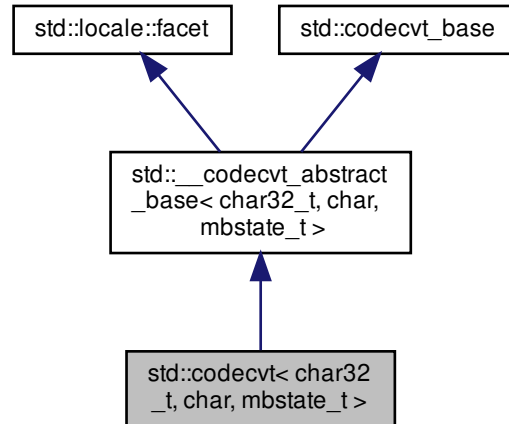
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.669 std::codecvt&lt; char32\_t, char, mbstate\_t &gt; Class Template Reference

Inheritance diagram for std::codecvt< char32\_t, char, mbstate\_t >:



## Public Types

- typedef char **extern\_type**
- typedef char32\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.669.1 Detailed Description

```
template<>
class std::codecvt< char32_t, char, mbstate_t >
```

Class codecvt<char32\_t, char, mbstate\_t> specialization.

Converts between UTF-32 and UTF-8.

Definition at line 525 of file codecvt.h.

#### 5.669.2 Member Function Documentation

5.669.2.1 `do_out()`

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

**See also**

`out` for more information.

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

5.669.2.2 `in()`

```
result std::__codecvt_abstract_base< char32_t , char , mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.



## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

5.669.2.3 `out()`

```
result std::__codecvt_abstract_base< char32_t , char , mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**5.669.2.4 unshift()**

```
result std::__codecvt_abstract_base< char32_t , char , mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

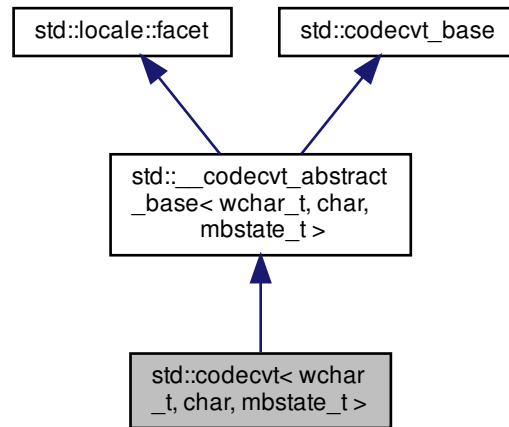
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.670 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



### Public Types

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## Friends

- class **messages< wchar\_t >**

## 5.670.1 Detailed Description

```
template<>
class std::codecvt< wchar_t, char, mbstate_t >
```

Class `codecvt<wchar_t, char, mbstate_t>` specialization.

Converts between narrow and wide characters in the native character set

Definition at line 401 of file `codecvt.h`.

## 5.670.2 Member Function Documentation

### 5.670.2.1 do\_out()

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const    [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

#### See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< wchar\_t, char, mbstate\_t >.

### 5.670.2.2 in()

```
result std::\_\_codecvt\_abstract\_base< wchar_t , char , mbstate_t >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const    [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling [codecvt::do\\_in](#).

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of [codecvt\\_base::result](#). If all the input is converted, returns [codecvt\\_base::ok](#). If no conversion is necessary, returns [codecvt\\_base::noconv](#). If the input ends early or there is insufficient space in the output, returns [codecvt\\_base::partial](#). Otherwise the conversion failed and [codecvt\\_base::error](#) is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

## 5.670.2.3 out()

```
result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**5.670.2.4 unshift()**

```
result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

codecvt\_base::result.

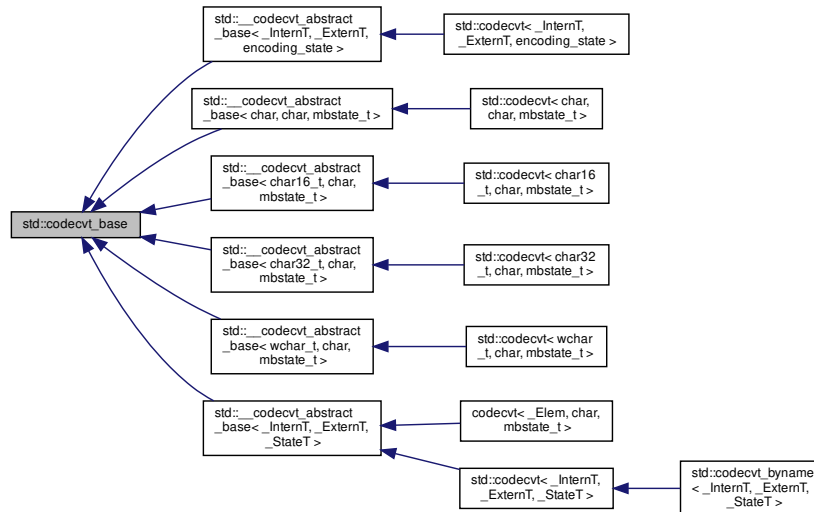
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.671 std::codecvt\_base Class Reference

Inheritance diagram for std::codecvt\_base:



## Public Types

- enum **result** { **ok**, **partial**, **error**, **noconv** }

## 5.671.1 Detailed Description

Empty base class for codecvt facet [22.2.1.5].

Definition at line 46 of file codecvt.h.

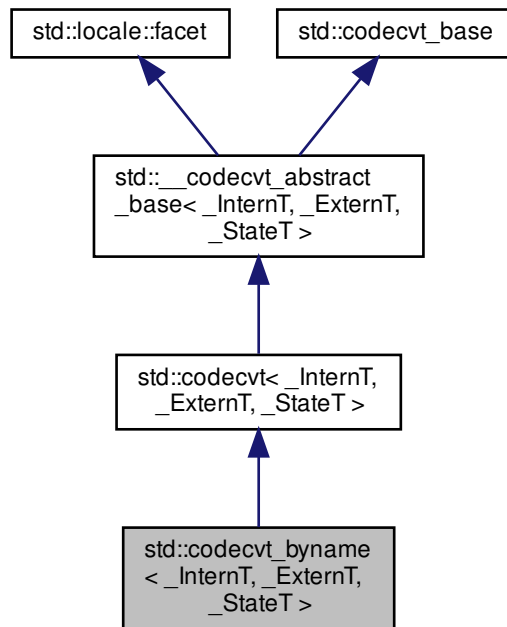
The documentation for this class was generated from the following file:

- [codecvt.h](#)



## 5.672 std::codecvt\_byname< \_InternT, \_ExternT, \_StateT > Class Template Reference

Inheritance diagram for std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >:



### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- **codecvt\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **codecvt\_byname** (const [string](#) & \_\_s, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* \_\_to\_next) const
- int **length** (state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_to\_next) const
- result **unshift** (state\_type & \_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_to\_next) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 5.672.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class codecvt\_byname [22.2.1.6].

Definition at line 582 of file codecvt.h.

## 5.672.2 Member Function Documentation

### 5.672.2.1 do\_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const    [protected], [virtual], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< `_InternT`, `_ExternT`, `_StateT` >.

### 5.672.2.2 in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::\_\_codecvt\_abstract\_base< _InternT, _ExternT, _StateT >::in (
    state_type & __state,
    const extern_type * __from,
    const extern_type * __from_end,
    const extern_type *& __from_next,
    intern_type * __to,
    intern_type * __to_end,
    intern_type *& __to_next ) const    [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

codecvt\_base::result.

Definition at line 196 of file codecvt.h.

## 5.672.2.3 out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
    state_type & __state,
    const intern_type * __from,
    const intern_type * __from_end,
    const intern_type *& __from_next,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

**Returns**

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

**5.672.2.4 unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
    state_type & __state,
    extern_type * __to,
    extern_type * __to_end,
    extern_type *& __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

## Returns

codecvt\_base::result.

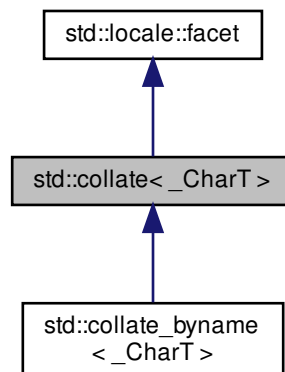
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.673 std::collate&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::collate<\_CharT>:



## Public Types

- typedef \_CharT [char\\_type](#)
- typedef [basic\\_string](#)<\_CharT> [string\\_type](#)

### Public Member Functions

- [collate](#) (size\_t \_\_refs=0)
- [collate](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<>  
int **\_M\_compare** (const char \*, const char \*) const throw()
- template<>  
int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- template<>  
size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<>  
size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- int [compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- long [hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- [string\\_type transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~collate](#) ()
- virtual int [do\\_compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_collate**

### 5.673.1 Detailed Description

```
template<typename _CharT>
class std::collate<_CharT>
```

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 642 of file locale\_classes.h.

### 5.673.2 Member Typedef Documentation

#### 5.673.2.1 char\_type

```
template<typename _CharT>
typedef _CharT std::collate<_CharT>::char_type
```

Public typedefs.

Definition at line 648 of file locale\_classes.h.

#### 5.673.2.2 string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::collate<_CharT>::string_type
```

Public typedefs.

Definition at line 649 of file locale\_classes.h.

### 5.673.3 Constructor & Destructor Documentation

#### 5.673.3.1 collate() [1/2]

```
template<typename _CharT>
std::collate<_CharT>::collate (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.



**Parameters**

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 669 of file `locale_classes.h`.

**5.673.3.2 `collate()`** [2/2]

```
template<typename _CharT>
std::collate< _CharT >::collate (
    __c_locale __cloc,
    size_t __refs = 0 ) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

<code>__cloc</code>	The C locale.
<code>__refs</code>	Passed to the base facet class.

Definition at line 683 of file `locale_classes.h`.

**5.673.3.3 `~collate()`**

```
template<typename _CharT>
virtual std::collate< _CharT >::~~collate ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 746 of file `locale_classes.h`.

**5.673.4 Member Function Documentation****5.673.4.1 `compare()`**

```
template<typename _CharT>
int std::collate< _CharT >::compare (
    const _CharT * __lo1,
    const _CharT * __hi1,
    const _CharT * __lo2,
    const _CharT * __hi2 ) const [inline]
```

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

## Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 700 of file `locale_classes.h`.

## 5.673.4.2 do\_compare()

```
template<typename _CharT>
int std::collate<_CharT>::do_compare (
    const _CharT * __lo1,
    const _CharT * __hi1,
    const _CharT * __lo2,
    const _CharT * __hi2 ) const [protected], [virtual]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

## See also

`compare()`.

## Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 161 of file `locale_classes.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

#### 5.673.4.3 do\_hash()

```
template<typename _CharT>
long std::collate< _CharT >::do_hash (
    const _CharT * __lo,
    const _CharT * __hi ) const [protected], [virtual]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

##### Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

##### Returns

Hash value.

Definition at line 256 of file locale\_classes.tcc.

#### 5.673.4.4 do\_transform()

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
    const _CharT * __lo,
    const _CharT * __hi ) const [protected], [virtual]
```

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

##### Parameters

<code>__lo</code>	Start.
<code>__hi</code>	End.

##### Returns

transformed string.

Definition at line 200 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

#### 5.673.4.5 hash()

```
template<typename _CharT>
long std::collate<_CharT>::hash (
    const _CharT * __lo,
    const _CharT * __hi ) const [inline]
```

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

##### Parameters

<code>↔ _lo</code>	Start of string.
<code>↔ _hi</code>	End of string.

##### Returns

Hash value.

Definition at line 733 of file locale\_classes.h.

#### 5.673.4.6 transform()

```
template<typename _CharT>
string_type std::collate<_CharT>::transform (
    const _CharT * __lo,
    const _CharT * __hi ) const [inline]
```

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

#### Parameters

<code>_↵ _lo</code>	Start of string.
<code>_↵ _hi</code>	End of string.

#### Returns

Transformed string\_type.

Definition at line 719 of file locale\_classes.h.

#### 5.673.5 Member Data Documentation

##### 5.673.5.1 id

```
template<typename _CharT>
locale::id std::collate< _CharT >::id [static]
```

Numpunct facet id.

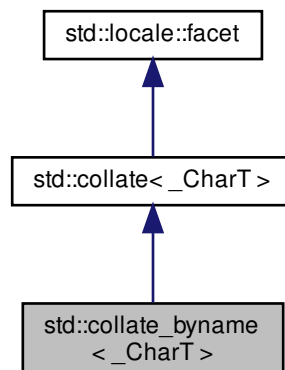
Definition at line 659 of file locale\_classes.h.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

#### 5.674 std::collate\_byname< \_CharT > Class Template Reference

Inheritance diagram for std::collate\_byname< \_CharT >:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string](#)< `_CharT` > [string\\_type](#)

## Public Member Functions

- **collate\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **collate\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- **int \_M\_compare** (const `_CharT` \*, const `_CharT` \*) const throw ()
- `template<>`  
**int \_M\_compare** (const char \*, const char \*) const throw()
- `template<>`  
**int \_M\_compare** (const `wchar_t` \*, const `wchar_t` \*) const throw()
- **size\_t \_M\_transform** (`_CharT` \*, const `_CharT` \*, size\_t) const throw ()
- `template<>`  
**size\_t \_M\_transform** (char \*, const char \*, size\_t) const throw()
- `template<>`  
**size\_t \_M\_transform** (`wchar_t` \*, const `wchar_t` \*, size\_t) const throw()
- **int compare** (const `_CharT` \*\_\_lo1, const `_CharT` \*\_\_hi1, const `_CharT` \*\_\_lo2, const `_CharT` \*\_\_hi2) const
- **long hash** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const
- **string\_type transform** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual **int do\_compare** (const `_CharT` \*\_\_lo1, const `_CharT` \*\_\_hi1, const `_CharT` \*\_\_lo2, const `_CharT` \*\_\_hi2) const
- virtual **long do\_hash** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const
- virtual **string\_type do\_transform** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const

## Static Protected Member Functions

- static `__c_locale` **\_S\_clone\_c\_locale** (`__c_locale` &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (`__c_locale` &\_\_cloc, const char \*\_\_s, `__c_locale` \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (`__c_locale` &\_\_cloc)
- static `__c_locale` **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static `__c_locale` **\_S\_lc\_ctype\_c\_locale** (`__c_locale` \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale __M_c_locale_collate`

### 5.674.1 Detailed Description

```
template<typename _CharT>
class std::collate_byname< _CharT >
```

class `collate_byname` [22.2.4.2].

Definition at line 816 of file `locale_classes.h`.

### 5.674.2 Member Typedef Documentation

#### 5.674.2.1 `char_type`

```
template<typename _CharT >
typedef _CharT std::collate_byname< _CharT >::char_type
```

Public typedefs.

Definition at line 821 of file `locale_classes.h`.

#### 5.674.2.2 `string_type`

```
template<typename _CharT >
typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type
```

Public typedefs.

Definition at line 822 of file `locale_classes.h`.

### 5.674.3 Member Function Documentation

#### 5.674.3.1 `compare()`

```
template<typename _CharT>
int std::collate< _CharT >::compare (
    const _CharT * __lo1,
    const _CharT * __hi1,
    const _CharT * __lo2,
    const _CharT * __hi2 ) const [inline], [inherited]
```

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

## Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

## Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 700 of file locale\_classes.h.

## 5.674.3.2 do\_compare()

```
template<typename _CharT>
int std::collate< _CharT >::do_compare (
    const _CharT * __lo1,
    const _CharT * __hi1,
    const _CharT * __lo2,
    const _CharT * __hi2 ) const [protected], [virtual], [inherited]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

## See also

compare().

## Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

## Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 161 of file locale\_classes.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::data(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::length().



#### 5.674.3.3 do\_hash()

```
template<typename _CharT>
long std::collate< _CharT >::do_hash (
    const _CharT * __lo,
    const _CharT * __hi ) const    [protected], [virtual], [inherited]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

##### Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

##### Returns

Hash value.

Definition at line 256 of file locale\_classes.tcc.

#### 5.674.3.4 do\_transform()

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
    const _CharT * __lo,
    const _CharT * __hi ) const    [protected], [virtual], [inherited]
```

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

##### Parameters

<code>__lo</code>	Start.
<code>__hi</code>	End.

##### Returns

transformed string.

Definition at line 200 of file locale\_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::length()`.

#### 5.674.3.5 hash()

```
template<typename _CharT>
long std::collate< _CharT >::hash (
    const _CharT * __lo,
    const _CharT * __hi ) const [inline], [inherited]
```

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

##### Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

##### Returns

Hash value.

Definition at line 733 of file locale\_classes.h.

#### 5.674.3.6 transform()

```
template<typename _CharT>
string_type std::collate< _CharT >::transform (
    const _CharT * __lo,
    const _CharT * __hi ) const [inline], [inherited]
```

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

**Parameters**

<code>_↵ _lo</code>	Start of string.
<code>_↵ _hi</code>	End of string.

**Returns**

Transformed string\_type.

Definition at line 719 of file locale\_classes.h.

**5.674.4 Member Data Documentation****5.674.4.1 id**

```
template<typename _CharT>
locale::id std::collate< _CharT >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 659 of file locale\_classes.h.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

**5.675 std::common\_type< \_Tp > Struct Template Reference**

Inherited by std::common\_type< \_Tp, \_Up, \_Vp... >.

**5.675.1 Detailed Description**

```
template<typename... _Tp>
struct std::common_type< _Tp >
```

common\_type

Definition at line 1930 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.676 std::complex&lt; \_Tp &gt; Struct Template Reference

## Public Types

- typedef `_Tp` `value_type`

## Public Member Functions

- constexpr `complex` (const `_Tp` &\_\_r=\_Tp(), const `_Tp` &\_\_i=\_Tp())
- constexpr `complex` (const `complex` &)=default
- template<typename `_Up` >  
constexpr `complex` (const `complex`< `_Up` > &\_\_z)
- constexpr `complex` `__rep` () const
- \_GLIBCXX\_ABI\_TAG\_CXX11 constexpr `_Tp` `imag` () const
- void `imag` (`_Tp` \_\_val)
- `complex`< `_Tp` > & `operator*=` (const `_Tp` &)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator*=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator+=` (const `_Tp` &\_\_t)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator+=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator-=` (const `_Tp` &\_\_t)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator-=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator/=` (const `_Tp` &)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator/=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator=` (const `_Tp` &)
- `complex` & `operator=` (const `complex` &)=default
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator=` (const `complex`< `_Up` > &)
- \_GLIBCXX\_ABI\_TAG\_CXX11 constexpr `_Tp` `real` () const
- void `real` (`_Tp` \_\_val)

## 5.676.1 Detailed Description

```
template<typename _Tp>
struct std::complex< _Tp >
```

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

## Parameters

<i>Tp</i>	Type of real and imaginary values.
-----------	------------------------------------

Definition at line 63 of file complex.

## 5.676.2 Member Typedef Documentation

### 5.676.2.1 value\_type

```
template<typename _Tp >
typedef _Tp std::complex< _Tp >::value_type
```

Value typedef.

Definition at line 125 of file complex.

## 5.676.3 Constructor & Destructor Documentation

### 5.676.3.1 complex() [1/2]

```
template<typename _Tp >
constexpr std::complex< _Tp >::complex (
    const _Tp & __r = _Tp(),
    const _Tp & __i = _Tp() ) [inline]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

Definition at line 129 of file complex.

### 5.676.3.2 complex() [2/2]

```
template<typename _Tp >
template<typename _Up >
constexpr std::complex< _Tp >::complex (
    const complex< _Up > & __z ) [inline]
```

Converting constructor.

Definition at line 139 of file complex.

## 5.676.4 Member Function Documentation

## 5.676.4.1 operator+=()

```
template<typename _Tp >
complex<_Tp>& std::complex< _Tp >::operator+= (
    const _Tp & __t ) [inline]
```

Add a scalar to this complex number.

Definition at line 184 of file complex.

## 5.676.4.2 operator-=()

```
template<typename _Tp >
complex<_Tp>& std::complex< _Tp >::operator-= (
    const _Tp & __t ) [inline]
```

Subtract a scalar from this complex number.

Definition at line 193 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

## 5.677 std::complex&lt; double &gt; Struct Template Reference

## Public Types

- typedef \_\_complex\_\_ double **\_ComplexT**
- typedef double **value\_type**

## Public Member Functions

- constexpr **complex** (\_ComplexT \_\_z)
- constexpr **complex** (double \_\_r=0.0, double \_\_i=0.0)
- constexpr **complex** (const [complex](#)< float > &\_\_z)
- constexpr **complex** (const [complex](#)< long double > &\_\_z)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double real() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double imag() const
- constexpr \_ComplexT **\_\_rep** () const
- void **imag** (double \_\_val)
- [complex](#) & **operator\*=** (double \_\_d)
- template<typename \_Tp >  
[complex](#) & **operator\*=** (const [complex](#)< \_Tp > &\_\_z)
- [complex](#) & **operator+=** (double \_\_d)
- template<typename \_Tp >  
[complex](#) & **operator+=** (const [complex](#)< \_Tp > &\_\_z)
- [complex](#) & **operator-=** (double \_\_d)
- template<typename \_Tp >  
[complex](#) & **operator-=** (const [complex](#)< \_Tp > &\_\_z)
- [complex](#) & **operator/=** (double \_\_d)
- template<typename \_Tp >  
[complex](#) & **operator/=** (const [complex](#)< \_Tp > &\_\_z)
- [complex](#) & **operator=** (double \_\_d)
- template<typename \_Tp >  
[complex](#) & **operator=** (const [complex](#)< \_Tp > &\_\_z)
- void **real** (double \_\_val)

### 5.677.1 Detailed Description

```
template<>
struct std::complex< double >
```

26.2.3 complex specializations `complex<double>` specialization

Definition at line 1221 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

### 5.678 `std::complex< float >` Struct Template Reference

#### Public Types

- typedef `__complex__ float` **\_ComplexT**
- typedef float **value\_type**

#### Public Member Functions

- constexpr **complex** (`_ComplexT __z`)
- constexpr **complex** (`float __r=0.0f, float __i=0.0f`)
- constexpr **complex** (`const complex< double > &`)
- constexpr **complex** (`const complex< long double > &`)
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr float `real()` const
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr float `imag()` const
- constexpr `_ComplexT __rep` () const
- void **imag** (`float __val`)
- [complex](#) & **operator\*=** (`float __f`)
- template<class `_Tp` >  
[complex](#) & **operator\*=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator+=** (`float __f`)
- template<typename `_Tp` >  
[complex](#) & **operator+=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator-=** (`float __f`)
- template<class `_Tp` >  
[complex](#) & **operator-=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator/=** (`float __f`)
- template<class `_Tp` >  
[complex](#) & **operator/=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator=** (`float __f`)
- template<typename `_Tp` >  
[complex](#) & **operator=** (`const complex< _Tp > &__z`)
- void **real** (`float __val`)

## 5.678.1 Detailed Description

```
template<>
struct std::complex< float >
```

26.2.3 complex specializations `complex<float>` specialization

Definition at line 1072 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

5.679 `std::complex< long double >` Struct Template Reference

## Public Types

- typedef `__complex__ long double` **\_ComplexT**
- typedef `long double` **value\_type**

## Public Member Functions

- constexpr **complex** (`_ComplexT __z`)
- constexpr **complex** (`long double __r=0.0L, long double __i=0.0L`)
- constexpr **complex** (`const complex< float > &__z`)
- constexpr **complex** (`const complex< double > &__z`)
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr `long double` **real**() const
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr `long double` **imag**() const
- constexpr `_ComplexT` **\_\_rep** () const
- void **imag** (`long double __val`)
- [complex](#) & **operator\*=** (`long double __r`)
- template<typename `_Tp` >  
[complex](#) & **operator\*=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator+=** (`long double __r`)
- template<typename `_Tp` >  
[complex](#) & **operator+=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator-=** (`long double __r`)
- template<typename `_Tp` >  
[complex](#) & **operator-=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator/=** (`long double __r`)
- template<typename `_Tp` >  
[complex](#) & **operator/=** (`const complex< _Tp > &__z`)
- [complex](#) & **operator=** (`long double __r`)
- template<typename `_Tp` >  
[complex](#) & **operator=** (`const complex< _Tp > &__z`)
- void **real** (`long double __val`)



### 5.679.1 Detailed Description

```
template<>
struct std::complex< long double >
```

### 26.2.3 complex specializations complex<long double> specialization

Definition at line 1371 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

## 5.680 std::condition\_variable Class Reference

### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

### Public Member Functions

- **condition\_variable** (const [condition\\_variable](#) &)=delete
- native\_handle\_type **native\_handle** ()
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable](#) & **operator=** (const [condition\\_variable](#) &)=delete
- void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock) noexcept
- template<typename \_Predicate >  
void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, \_Predicate \_\_p)
- template<typename \_Rep, typename \_Period >  
[cv\\_status](#) **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Rep, typename \_Period, typename \_Predicate >  
bool **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Duration >  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_\_clock\_t, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration >  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration, typename \_Predicate >  
bool **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

## 5.680.1 Detailed Description

`condition_variable`

Definition at line 65 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

5.681 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference

## Public Types

- `typedef _Iftrue` **type**

## 5.681.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
struct std::conditional< _Cond, _Iftrue, _Iffalse >
```

Define a member typedef `type` to one of two argument types.

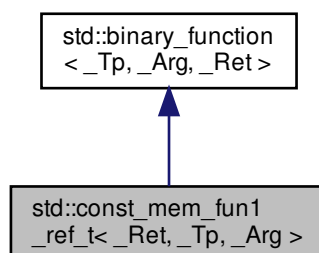
Definition at line 92 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.682 `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

Inheritance diagram for `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- **const\_mem\_fun1\_ref\_t** (`_Ret`(`_Tp`::\*`__pf`)(`_Arg`) const)
- `_Ret` **operator()** (const `_Tp` &`__r`, `_Arg` `__x`) const

### 5.682.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1305 of file `stl_function.h`.

### 5.682.2 Member Typedef Documentation

#### 5.682.2.1 first\_argument\_type

```
typedef _Tp std::binary\_function< _Tp , _Arg , _Ret >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.682.2.2 result\_type

```
typedef _Ret std::binary\_function< _Tp , _Arg , _Ret >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

## 5.682.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

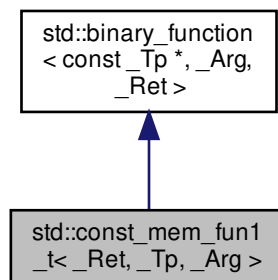
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.683 std::const\_mem\_fun1\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef const \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

## Public Member Functions

- **const\_mem\_fun1\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg) const)
- \_Ret **operator()** (const \_Tp \* \_\_p, \_Arg \_\_x) const

### 5.683.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1269 of file `stl_function.h`.

### 5.683.2 Member Typedef Documentation

#### 5.683.2.1 first\_argument\_type

```
typedef const _Tp * std::binary_function< const _Tp * , _Arg , _Ret >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.683.2.2 result\_type

```
typedef _Ret std::binary_function< const _Tp * , _Arg , _Ret >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 5.683.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< const _Tp * , _Arg , _Ret >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

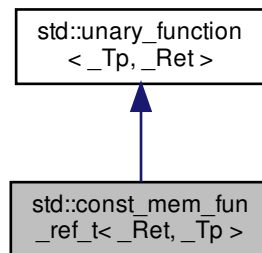
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.684 std::const\_mem\_fun\_ref\_t&lt; \_Ret, \_Tp &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >:



## Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

## Public Member Functions

- **const\_mem\_fun\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)() const)
- \_Ret **operator()** (const \_Tp &\_\_r) const

## 5.684.1 Detailed Description

```
template<typename _Ret, typename _Tp>  
class std::const_mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

Definition at line 1233 of file stl\_function.h.

## 5.684.2 Member Typedef Documentation

#### 5.684.2.1 `argument_type`

```
typedef _Tp std::unary_function< _Tp , _Ret >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.684.2.2 `result_type`

```
typedef _Ret std::unary_function< _Tp , _Ret >::result_type [inherited]
```

`result_type` is the return type

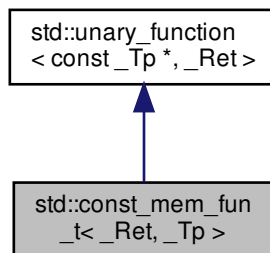
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.685 `std::const_mem_fun_t< _Ret, _Tp >` Class Template Reference

Inheritance diagram for `std::const_mem_fun_t< _Ret, _Tp >`:



#### Public Types

- `typedef const _Tp * argument\_type`
- `typedef _Ret result\_type`

## Public Member Functions

- **const\_mem\_fun\_t** (\_Ret(\_Tp::\*\_\_pf)() const)
- **\_Ret operator()** (const \_Tp \*\_\_p) const

### 5.685.1 Detailed Description

```
template<typename _Ret, typename _Tp>  
class std::const_mem_fun_t<_Ret,_Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 1197 of file stl\_function.h.

### 5.685.2 Member Typedef Documentation

#### 5.685.2.1 argument\_type

```
typedef const _Tp * std::unary_function< const _Tp * , _Ret >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

#### 5.685.2.2 result\_type

```
typedef _Ret std::unary_function< const _Tp * , _Ret >::result_type [inherited]
```

result\_type is the return type

Definition at line 111 of file stl\_function.h.

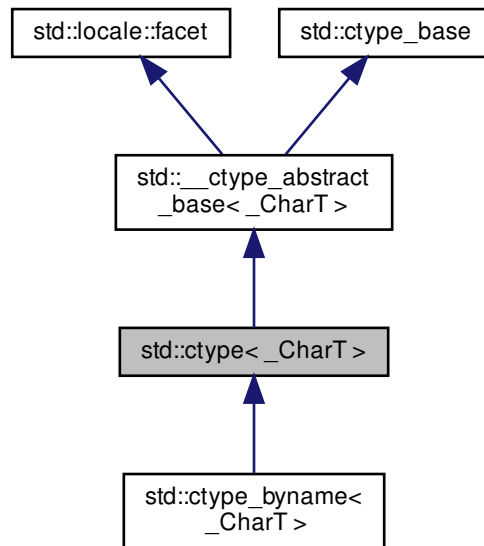
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)



## 5.686 std::ctype<\_CharT> Class Template Reference

Inheritance diagram for std::ctype<\_CharT>:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef `__ctype_abstract_base<_CharT>::mask` **mask**

### Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char\_type \_\_c) const
- const char\_type \* **is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \*\_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \*\_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \*\_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#), [char](#) \_\_dfault) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_dfault, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_dest) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale](#) [\\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc, const [char](#) \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale](#) [\\_S\\_get\\_c\\_locale](#) ()
- static const [char](#) \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale](#) [\\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const [char](#) \* \_\_s)

### 5.686.1 Detailed Description

```
template<typename _CharT>
class std::ctype<_CharT>
```

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in `__ctype_abstract_base`, to allow for implementation flexibility. See `ctype<wchar_t>` for an example. The functions are documented in `__ctype_abstract_base`.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 612 of file `locale_facets.h`.

### 5.686.2 Member Function Documentation

#### 5.686.2.1 `do_is()` [1/2]

```
template<typename _CharT>
virtual bool std::ctype<_CharT>::do_is (
    mask __m,
    char_type __c ) const [protected], [virtual]
```

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

<code>__c</code>	The <code>char_type</code> to find the mask of.
<code>__m</code>	The mask to compare against.

#### Returns

`(M & __m) != 0`.

Implements `std::__ctype_abstract_base<_CharT>`.

### 5.686.2.2 do\_is() [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

### 5.686.2.3 do\_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype<_CharT>::do_narrow (
    char_type __c,
    char __default ) const [protected], [virtual]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, default is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
<code>__default</code>	Char to return if conversion fails.

**Returns**

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::narrow()`.

**5.686.2.4 do\_narrow()** [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to ) const [protected], [virtual]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

## 5.686.2.5 do\_scan\_is()

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

## Returns

Pointer to a matching char\_type if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

## 5.686.2.6 do\_scan\_not()

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

## Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a non-matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.686.2.7 do\_tolower()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
    char_type __c ) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::tolower()`.

**5.686.2.8 do\_tolower()** [2/2]



```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_tolower (
    char_type * __lo,
    const char_type * __hi ) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

## Parameters

<a href="#"></a> <code>_lo</code>	Pointer to start of range.
<a href="#"></a> <code>_hi</code>	Pointer to end of range.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

## 5.686.2.9 do\_toupper() [1/2]


```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_toupper (
    char_type __c ) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

<a href="#"></a> <code>__c</code>	The <code>char_type</code> to convert.
---	--

## Returns

The uppercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::toupper()`.

## 5.686.2.10 do\_toupper() [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_toupper (
```



```
char_type * __lo,  
const char_type * __hi ) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT >](#).

#### 5.686.2.11 do\_widen() [1/2]

```
template<typename _CharT>  
virtual char_type std::ctype<_CharT >::do_widen (  
    char __c ) const [protected], [virtual]
```

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The converted `char_type`

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::widen()`.

#### 5.686.2.12 do\_widen() [2/2]

```
template<typename _CharT>
virtual const char* std::ctype<\_CharT>::do\_widen (
    const char * __lo,
    const char * __hi,
    char\_type * __to ) const    [protected], [virtual]
```

Widen char array.

This function converts each char in the input to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<a href="#">_↵ _lo</a>	Pointer to start range.
<a href="#">_↵ _hi</a>	Pointer to end of range.
<a href="#">_↵ _to</a>	Pointer to the destination array.

#### Returns

[\\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 5.686.2.13 is() [1/2]

```
template<typename _CharT>
bool std::\_\_ctype\_abstract\_base<\_CharT>::is (
    mask __m,
    char\_type __c ) const    [inline], [inherited]
```

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵  
type>::do_is()`.

**Parameters**

<code>↵ _c</code>	The char_type to compare the mask of.
<code>↵ _m</code>	The mask to compare against.

**Returns**

(M & \_\_m) != 0.

Definition at line 169 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter >::get()`.

**5.686.2.14 is() [2/2]**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

**Returns**

`__hi`.

Definition at line 186 of file locale\_facets.h.

**5.686.2.15 narrow() [1/2]**

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT >::narrow (
```

```
char_type __c,
char __dfault ) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 331 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_Inlter>::get(), and std::time\_put<\_CharT, \_Outlter>::put().

#### 5.686.2.16 narrow() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to ) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_dfault, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 353 of file `locale_facets.h`.

**5.686.2.17 scan\_is()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to matching `char_type` if found, else `__hi`.

Definition at line 202 of file `locale_facets.h`.




**5.686.2.18 scan\_not()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

## Parameters

 _m	The mask to compare against.
 _lo	Pointer to first char in range.
 _hi	Pointer to end of range.

## Returns

Pointer to non-matching char if found, else \_\_hi.

Definition at line 218 of file locale\_facets.h.


## 5.686.2.19 tolower() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::tolower (
    char_type __c ) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

## Parameters

 _c	The char_type to convert.
---	---------------------------

## Returns

The lowercase char\_type if convertible, else \_\_c.

Definition at line 261 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_InIter>::get().

## 5.686.2.20 tolower() [2/2]



```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::tolower (
```

```
char_type * __lo,  
const char_type * __hi ) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

## Parameters

 <code>__lo</code>	Pointer to start of range.
 <code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 276 of file locale\_facets.h.


## 5.686.2.21 toupper() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::toupper (
    char_type __c ) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

## Parameters

 <code>__c</code>	The char_type to convert.
--	---------------------------

## Returns

The uppercase char\_type if convertible, else `__c`.

Definition at line 232 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_Inlter>::get().

## 5.686.2.22 toupper() [2/2]



```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::toupper (
    char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).



**Parameters**

<a href="#"></a> <code>_lo</code>	Pointer to start of range.
<a href="#"></a> <code>_hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 247 of file `locale_facets.h`.

**5.686.2.23 `widen()`** [1/2]


```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
    char __c ) const [inline], [inherited]
```

Widen char to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<a href="#"></a> <code>_c</code>	The char to convert.
--	----------------------

**Returns**

The converted `char_type`.

Definition at line 293 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::time_put< _CharT, _Outlter >::do_put()`, and `std::money_put< _CharT, _Outlter >::do_put()`.

**5.686.2.24 `widen()`** [2/2]

```
template<typename _CharT>
const char* std::__ctype_abstract_base< _CharT >::widen (
```

```
const char * __lo,  
const char * __hi,  
char_type * __to ) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<a href="#"><code>__lo</code></a>	Pointer to start of range.
<a href="#"><code>__hi</code></a>	Pointer to end of range.
<a href="#"><code>__to</code></a>	Pointer to the destination array.

#### Returns

[`\_\_hi`](#).

Definition at line 312 of file locale\_facets.h.

### 5.686.3 Member Data Documentation

#### 5.686.3.1 id

```
template<typename _CharT>  
locale::id std::ctype<_CharT>::id [static]
```

The facet id for ctype<char\_type>

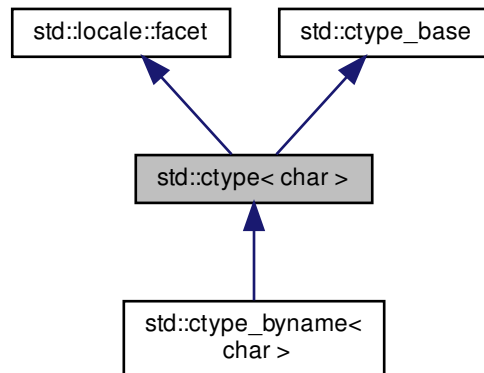
Definition at line 620 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.687 std::ctype< char > Class Template Reference

Inheritance diagram for std::ctype< char >:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef char **char\_type**
- typedef unsigned short **mask**

### Public Member Functions

- **ctype** (const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- **ctype** (\_\_c\_locale \_\_cloc, const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char \_\_c) const
- const char \* **is** (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* **scan\_is** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* **scan\_not** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* **table** () const throw ()
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual [~ctype](#) ()
- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault \_\_attribute\_\_((\_\_unused\_\_))) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault \_\_attribute\_\_((\_\_unused\_\_)), char \* \_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- `__c_locale __M_c_locale_ctype`
- `bool __M_del`
- `char __M_narrow [1+static_cast< unsigned char >(-1)]`
- `char __M_narrow_ok`
- `const mask * __M_table`
- `__to_type __M_tolower`
- `__to_type __M_toupper`
- `char __M_widen [1+static_cast< unsigned char >(-1)]`
- `char __M_widen_ok`

### 5.687.1 Detailed Description

```
template<>
class std::ctype< char >
```

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

Definition at line 681 of file `locale_facets.h`.

### 5.687.2 Member Typedef Documentation

#### 5.687.2.1 `char_type`

```
typedef char std::ctype< char >::char_type
```

Typedef for the template parameter `char`.

Definition at line 686 of file `locale_facets.h`.

### 5.687.3 Constructor & Destructor Documentation

#### 5.687.3.1 `ctype()` [1/2]

```
std::ctype< char >::ctype (
    const mask * __table = 0,
    bool __del = false,
    size_t __refs = 0 ) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

<code>__table</code>	If non-zero, table is used as the per-char mask. Else <code>classic_table()</code> is used.
<code>__del</code>	If true, passes ownership of table to this facet.
<code>__refs</code>	Passed to the base facet class.

## 5.687.3.2 ctype() [2/2]

```
std::ctype< char >::ctype (
    __c_locale __cloc,
    const mask * __table = 0,
    bool __del = false,
    size_t __refs = 0 ) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

## Parameters

<code>__cloc</code>	Handle to C locale data.
<code>__table</code>	If non-zero, table is used as the per-char mask.
<code>__del</code>	If true, passes ownership of table to this facet.
<code>__refs</code>	Passed to the base facet class.

## 5.687.3.3 ~ctype()

```
virtual std::ctype< char >::~~ctype ( ) [protected], [virtual]
```

Destructor.

This function deletes `table()` if `del` was true in the constructor.

## 5.687.4 Member Function Documentation

## 5.687.4.1 classic\_table()

```
static const mask* std::ctype< char >::classic_table ( ) throw ( ) [static]
```

Returns a pointer to the C locale mask table.

**5.687.4.2** `do_narrow()` [1/2]

```
virtual char std::ctype< char >::do_narrow (
    char_type __c,
    char __default __attribute__((unused)) ) const [inline], [protected], [virtual]
```

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *default* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The char to convert.
<code>__default</code>	Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 1134 of file `locale_facets.h`.

**5.687.4.3** `do_narrow()` [2/2]

```
virtual const char_type* std::ctype< char >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default __attribute__((unused)),
    char * __to ) const [inline], [protected], [virtual]
```

Narrow char array to char array.

This virtual function converts each char in the range `[lo,hi)` to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**`__hi`.

Definition at line 1160 of file locale\_facets.h.

**5.687.4.4 do\_tolower()** [1/2]

```
virtual char_type std::ctype< char >::do_tolower (
    char_type __c ) const    [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**The lowercase char if convertible, else `__c`.**5.687.4.5 do\_tolower()** [2/2]

```
virtual const char_type* std::ctype< char >::do_tolower (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.



**Returns**

`__hi`.

**5.687.4.6 do\_toupper()** [1/2]

```
virtual char_type std::ctype< char >::do_toupper (
    char_type __c ) const    [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The uppercase char if convertible, else `__c`.

**5.687.4.7 do\_toupper()** [2/2]

```
virtual const char_type* std::ctype< char >::do_toupper (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**`__hi.`**5.687.4.8 do\_widen()** [1/2]

```
virtual char_type std::ctype< char >::do_widen (
    char __c ) const    [inline], [protected], [virtual]
```

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The converted character.

Definition at line 1084 of file locale\_facets.h.

**5.687.4.9 do\_widen()** [2/2]

```
virtual const char* std::ctype< char >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to ) const    [inline], [protected], [virtual]
```




Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

<a href="#"></a> <code>__lo</code>	Pointer to start of range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.
<a href="#"></a> <code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 1107 of file locale\_facets.h.



**5.687.4.10 is()** [1/2]

```
bool std::ctype< char >::is (
    mask __m,
    char __c ) const [inline]
```

Test char classification.

This function compares the mask table[c] to `__m`.

**Parameters**

<a href="#"></a> <code>__c</code>	The char to compare the mask of.
<a href="#"></a> <code>__m</code>	The mask to compare against.

**Returns**

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file ctype\_inline.h.

**5.687.4.11 is()** [2/2]

```
const char * std::ctype< char >::is (
    const char * __lo,
    const char * __hi,
    mask * __vec ) const [inline]
```

Return a mask array.

This function finds the mask for each char in the range `[lo, hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

**Returns**

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

**5.687.4.12 narrow()** [1/2]

```
char std::ctype< char >::narrow (
    char_type __c,
    char __dfault ) const [inline]
```

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an undervied `ctype<char>` facet, `c` will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

**Parameters**

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

**Returns**

The converted character.

Definition at line 931 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_narrow()`.

## 5.687.4.13 narrow() [2/2]

```
const char_type* std::ctype< char >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to ) const [inline]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, default, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 964 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_narrow().

## 5.687.4.14 scan\_is()

```
const char * std::ctype< char >::scan_is (
    mask __m,
    const char * __lo,
    const char * __hi ) const [inline]
```

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file `ctype_inline.h`.

**5.687.4.15 `scan_not()`**

```
const char * std::ctype< char >::scan_not (
    mask __m,
    const char * __lo,
    const char * __hi ) const [inline]
```

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [`__lo`,`__hi`) for which `is(m,char)` is false.

**Parameters**

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file `ctype_inline.h`.

## 5.687.4.16 table()

```
const mask* std::ctype< char >::table ( ) const throw ( )    [inline]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic\_table() if none was provided.

Definition at line 983 of file locale\_facets.h.

## 5.687.4.17 tolower() [1/2]


```
char_type std::ctype< char >::tolower (
    char_type __c ) const    [inline]
```

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_c). do\_tolower() must always return the same result for the same input.

## Parameters

 __c	The char to convert.
---	----------------------

## Returns

The lowercase char if convertible, else \_\_c.

Definition at line 835 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_tolower().

## 5.687.4.18 tolower() [2/2]

```
const char_type* std::ctype< char >::tolower (
    char_type * __lo,
    const char_type * __hi ) const    [inline]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_lo, \_\_hi). do\_tolower() must always return the same result for the same input.



**Parameters**

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 852 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_tolower()`.

**5.687.4.19 toupper()** [1/2]

```
char_type std::ctype< char >::toupper (
    char_type __c ) const [inline]
```

Convert to uppercase.

This function converts the `char` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The uppercase char if convertible, else `__c`.

Definition at line 802 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_toupper()`.

**5.687.4.20 toupper()** [2/2]

```
const char_type* std::ctype< char >::toupper (
    char_type * __lo,
    const char_type * __hi ) const [inline]
```

Convert array to uppercase.

This function converts each char in the range [`__lo`,`__hi`) to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(__lo, __hi)`. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Definition at line 819 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_toupper()`.

#### 5.687.4.21 widen() [1/2]

```
char_type std::ctype< char >::widen (
    char __c ) const [inline]
```

Widen char.

This function converts the `char` to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

#### Returns

The converted character.

Definition at line 872 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

#### 5.687.4.22 `widen()` [2/2]

```
const char* std::ctype< char >::widen (  
    const char * __lo,  
    const char * __hi,  
    char_type * __to ) const [inline]
```

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

##### Parameters

<code>↔ __lo</code>	Pointer to first char in range.
<code>↔ __hi</code>	Pointer to end of range.
<code>↔ __to</code>	Pointer to the destination array.

##### Returns

`__hi`.

Definition at line 899 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

#### 5.687.5 Member Data Documentation

##### 5.687.5.1 `id`

```
locale::id std::ctype< char >::id [static]
```

The facet id for `ctype<char>`

Definition at line 703 of file `locale_facets.h`.

## 5.687.5.2 table\_size

```
const size_t std::ctype< char >::table_size [static]
```

The size of the mask table. It is SCHAR\_MAX + 1.

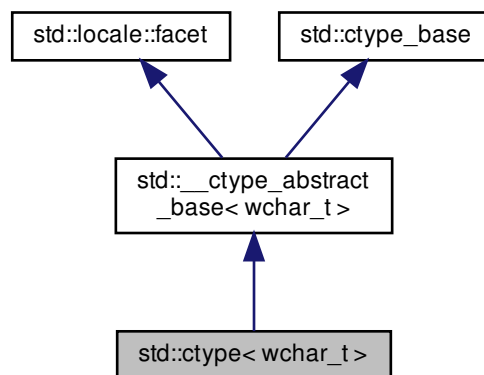
Definition at line 705 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

## 5.688 std::ctype&lt; wchar\_t &gt; Class Template Reference

Inheritance diagram for std::ctype< wchar\_t >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef wctype\_t **\_\_wmask\_type**
- typedef wchar\_t **char\_type**
- typedef unsigned short **mask**

### Public Member Functions

- [ctype](#) (size\_t \_\_refs=0)
- [ctype](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool [is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const [char\\_type](#) \* [scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual [~ctype](#) ()
- [\\_\\_wmask\\_type](#) **\_M\_convert\_to\_wmask** (const mask \_\_m) const throw ()
- void **\_M\_initialize\_ctype** () throw ()
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual char [do\\_narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

**Static Protected Member Functions**

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

**Protected Attributes**

- mask **\_M\_bit** [16]
- \_\_c\_locale **\_M\_c\_locale\_ctype**
- char **\_M\_narrow** [128]
- bool **\_M\_narrow\_ok**
- wint\_t **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- \_\_wmask\_type **\_M\_wmask** [16]

**5.688.1 Detailed Description**

template<>

class std::ctype< wchar\_t >

The ctype<wchar\_t> specialization.

This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.

ctype<wchar\_t> inherits its public methods from \_\_ctype\_abstract\_base<wchar\_t>.

Definition at line 1186 of file locale\_facets.h.

**5.688.2 Member Typedef Documentation****5.688.2.1 char\_type**

```
typedef wchar_t std::ctype< wchar_t >::char_type
```

Typedef for the template parameter wchar\_t.

Definition at line 1191 of file locale\_facets.h.

**5.688.3 Constructor & Destructor Documentation****5.688.3.1 ctype() [1/2]**

```
std::ctype< wchar_t >::ctype (
    size_t __refs = 0 ) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

**Parameters**

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

**5.688.3.2 ctype()** [2/2]

```
std::ctype< wchar_t >::ctype (
    __c_locale __cloc,
    size_t __refs = 0 ) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

**Parameters**

<code>__cloc</code>	Handle to C locale data.
<code>__refs</code>	Passed to the base facet class.

**5.688.3.3 ~ctype()**

```
virtual std::ctype< wchar_t >::~ctype ( ) [protected], [virtual]
```

Destructor.

**5.688.4 Member Function Documentation****5.688.4.1 do\_is()** [1/2]

```
virtual bool std::ctype< wchar_t >::do_is (
    mask __m,
    char_type __c ) const [protected], [virtual]
```

Test `wchar_t` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

## Parameters

<a href="#"><code>__c</code></a>	The <code>wchar_t</code> to find the mask of.
<a href="#"><code>__m</code></a>	The mask to compare against.

## Returns

`(M & __m) != 0.`

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

## 5.688.4.2 do\_is() [2/2]

```
virtual const char_type* std::ctype< wchar_t >::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each `wchar_t` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

## Parameters

<a href="#"><code>__lo</code></a>	Pointer to start of range.
<a href="#"><code>__hi</code></a>	Pointer to end of range.
<a href="#"><code>__vec</code></a>	Pointer to an array of mask storage.

## Returns

[`\_\_hi`](#).

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

## 5.688.4.3 do\_narrow() [1/2]

```
virtual char std::ctype< wchar_t >::do_narrow (
    char_type __c,
    char __default ) const [protected], [virtual]
```



Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an undervied `ctype<wchar_t>` facet, `c` will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__c</code>	The <code>wchar_t</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

#### Returns

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### 5.688.4.4 `do_narrow()` [2/2]

```
virtual const char_type* std::ctype< wchar_t >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to ) const [protected], [virtual]
```

Narrow `wchar_t` array to `char` array.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `wchar_t` in the input that cannot be converted, `dfault` is used instead. For an undervied `ctype<wchar_t>` facet, the argument will be copied, casting each element to `char`.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**`__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).**5.688.4.5 do\_scan\_is()**

```
virtual const char_type* std::ctype< wchar_t >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [virtual]
```

Find `wchar_t` matching mask.This function searches for and returns the first `wchar_t` `c` in `[__lo,__hi)` for which `is(__m,c)` is true.`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.**Parameters**




<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**Pointer to a matching `wchar_t` if found, else `__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).**5.688.4.6 do\_scan\_not()**

```
virtual const char_type* std::ctype< wchar_t >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [virtual]
```

Find `wchar_t` not matching mask.This function searches for and returns a pointer to the first `wchar_t` `c` of `[__lo,__hi)` for which `is(__m,c)` is false.`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

<a href="#"></a> <code>__m</code>	The mask to compare against.
<a href="#"></a> <code>__lo</code>	Pointer to start of range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a non-matching `wchar_t` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.688.4.7 do\_tolower()** [1/2]


```
virtual char\_type std::ctype< wchar_t >::do_tolower (  
    char\_type __c ) const    [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

<a href="#"></a> <code>__c</code>	The <code>wchar_t</code> to convert.
---	--------------------------------------

**Returns**

The lowercase `wchar_t` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.688.4.8 do\_tolower()** [2/2]

```
virtual const char\_type* std::ctype< wchar_t >::do_tolower (  
    char\_type * __lo,  
    const char\_type * __hi ) const    [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

<code>_↔ _lo</code>	Pointer to start of range.
<code>_↔ _hi</code>	Pointer to end of range.

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### 5.688.4.9 do\_toupper() [1/2]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
    char_type __c ) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

<code>_↔ _c</code>	The <code>wchar_t</code> to convert.
------------------------	--------------------------------------

#### Returns

The uppercase `wchar_t` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.688.4.10 do\_toupper()** [2/2]

```
virtual const char_type* std::ctype< wchar_t >::do_toupper (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

**5.688.4.11 do\_widen()** [1/2]

```
virtual char_type std::ctype< wchar_t >::do_widen (
    char __c ) const    [protected], [virtual]
```

Widen char to `wchar_t`.

This virtual function converts the `char` to `wchar_t` using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be cast to `wchar_t`.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The converted wchar\_t.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.688.4.12 do\_widen()** [2/2]

```
virtual const char* std::ctype< wchar_t >::do_widen (
    const char * __lo,
    const char * __hi,
    char_type * __to ) const [protected], [virtual]
```




Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be copied, casting each element to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

<a href="#"> __lo</a>	Pointer to start range.
<a href="#"> __hi</a>	Pointer to end of range.
<a href="#"> __to</a>	Pointer to the destination array.

**Returns**

[!\[\]\(aab88c0d099e5d18d6533a97b13ec28d\_img.jpg\) \\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.688.4.13 is()** [1/2]

```
bool std::__ctype_abstract_base< wchar_t >::is (
    mask __m,
    char_type __c ) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_[!\[\]\(f9f168a9979beed8b01f8750d577d508\_img.jpg\)](#)type>::do\_is().

**Parameters**

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

**Returns**

(M & \_\_m) != 0.

Definition at line 169 of file locale\_facets.h.

**5.688.4.14 is()** [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

**Returns**

`__hi`.

Definition at line 186 of file locale\_facets.h.

**5.688.4.15 narrow()** [1/2]

```
char std::__ctype_abstract_base< wchar_t >::narrow (
    char_type __c,
    char __default ) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__default</code>	Char to return if conversion fails.

## Returns

The converted char.

Definition at line 331 of file `locale_facets.h`.

## 5.688.4.16 narrow() [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to ) const [inline], [inherited]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 353 of file `locale_facets.h`.

## 5.688.4.17 scan\_is()

```
const char_type* std::__ctype_abstract_base< wchar_t >::scan_is (
    mask __m,
```






```
const char_type * __lo,  
const char_type * __hi ) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

#### Parameters

 __m	The mask to compare against.
 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

#### Returns

Pointer to matching char\_type if found, else \_\_hi.

Definition at line 202 of file locale\_facets.h.




#### 5.688.4.18 scan\_not()

```
const char_type* std::__ctype_abstract_base< wchar_t >::scan_not (  
    mask __m,  
    const char_type * __lo,  
    const char_type * __hi ) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

#### Parameters

 __m	The mask to compare against.
 __lo	Pointer to first char in range.
 __hi	Pointer to end of range.

#### Returns

Pointer to non-matching char if found, else \_\_hi.

Definition at line 218 of file locale\_facets.h.


#### 5.688.4.19 tolower() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::tolower (
    char_type __c ) const    [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

##### Parameters

 __c	The char_type to convert.
--	---------------------------

##### Returns

The lowercase char\_type if convertible, else \_\_c.

Definition at line 261 of file locale\_facets.h.



#### 5.688.4.20 tolower() [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::tolower (
    char_type * __lo,
    const char_type * __hi ) const    [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

##### Parameters

 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

##### Returns

\_\_hi.

Definition at line 276 of file locale\_facets.h.


#### 5.688.4.21 toupper() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::toupper (
    char_type __c ) const    [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

##### Parameters

 __c	The char_type to convert.
--	---------------------------

##### Returns

The uppercase char\_type if convertible, else \_\_c.

Definition at line 232 of file locale\_facets.h.



#### 5.688.4.22 toupper() [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::toupper (
    char_type * __lo,
    const char_type * __hi ) const    [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

##### Parameters

 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

##### Returns

\_\_hi.

Definition at line 247 of file locale\_facets.h.

#### 5.688.4.23 widen() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::widen (  
    char __c ) const    [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

##### Parameters

$\leftrightarrow$ __c	The char to convert.
--------------------------	----------------------

##### Returns

The converted char\_type.

Definition at line 293 of file locale\_facets.h.

#### 5.688.4.24 widen() [2/2]

```
const char* std::__ctype_abstract_base< wchar_t >::widen (  
    const char * __lo,  
    const char * __hi,  
    char_type * __to ) const    [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

##### Parameters

$\leftrightarrow$ __lo	Pointer to start of range.
$\leftrightarrow$ __hi	Pointer to end of range.
$\leftrightarrow$ __to	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 312 of file `locale_facets.h`.

## 5.688.5 Member Data Documentation

5.688.5.1 `id`

```
locale::id std::ctype< wchar_t >::id [static]
```

The facet id for `ctype<wchar_t>`

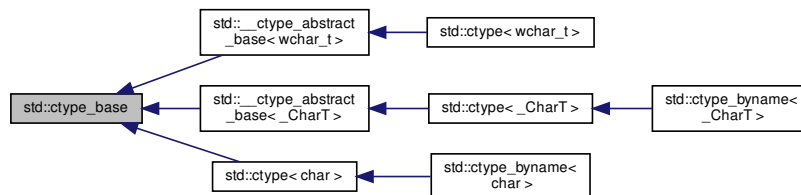
Definition at line 1209 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

5.689 `std::ctype_base` Struct Reference

Inheritance diagram for `std::ctype_base`:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## 5.689.1 Detailed Description

Base class for ctype.

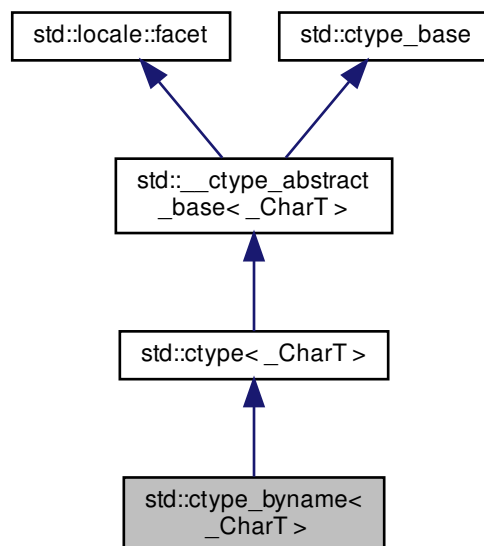
Definition at line 41 of file ctype\_base.h.

The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

## 5.690 std::ctype\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::ctype\_byname<\_CharT>:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef ctype< \_CharT >::mask **mask**

### Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **ctype\_byname** (const string & \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char\_type \_\_c) const
- const char\_type \* **is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \*\_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \*\_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \*\_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const
- virtual const char\_type \* **do\_is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \*\_\_vec) const
- virtual char **do\_narrow** (char\_type, char \_\_dfault) const
- virtual const char\_type \* **do\_narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \*\_\_to) const
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_tolower** (char\_type \_\_c) const
- virtual const char\_type \* **do\_tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_toupper** (char\_type \_\_c) const
- virtual const char\_type \* **do\_toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_widen** (char \_\_c) const
- virtual const char \* **do\_widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \*\_\_dest) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.690.1 Detailed Description

```
template<typename _CharT>
class std::ctype_byname<_CharT>
```

class ctype\_byname [22.2.1.2].

Definition at line 1478 of file locale\_facets.h.

## 5.690.2 Member Function Documentation

## 5.690.2.1 do\_is() [1/2]



```
template<typename _CharT>
virtual bool std::ctype<_CharT>::do_is (
    mask __m,
    char_type __c ) const [protected], [virtual], [inherited]
```

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

 <b>__c</b>	The char_type to find the mask of.
 <b>__m</b>	The mask to compare against.

## Returns

(*M* & *\_\_m*) != 0.



Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 5.690.2.2 do\_is() [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype<_CharT>::do_is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const    [protected], [virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

##### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

##### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 5.690.2.3 do\_narrow() [1/2]

```
template<typename _CharT>
virtual char std::__ctype<_CharT>::do_narrow (
    char_type __c,
    char __dfault ) const    [protected], [virtual], [inherited]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__default</code>	Char to return if conversion fails.

## Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::narrow()`.

## 5.690.2.4 do\_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default,
    char * __to ) const [protected], [virtual], [inherited]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

### 5.690.2.5 do\_scan\_is()

```
template<typename _CharT>
virtual const char_type* std::ctype< _CharT >::do_scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [virtual], [inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

#### Returns

Pointer to a matching char\_type if found, else \_\_hi.

Implements `std::__ctype_abstract_base< _CharT >`.

### 5.690.2.6 do\_scan\_not()

```
template<typename _CharT>
virtual const char_type* std::ctype< _CharT >::do_scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [protected], [virtual], [inherited]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

**Returns**

Pointer to a non-matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.690.2.7 do\_tolower()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_tolower (
    char_type __c ) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

<code>__c</code>	The char_type to convert.
------------------	---------------------------

**Returns**

The lowercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::tolower()`.

**5.690.2.8 do\_tolower()** [2/2]



```
template<typename _CharT>
virtual const char_type* std::ctype< _CharT >::do_tolower (
    char_type * __lo,
    const char_type * __hi ) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

<a href="#"></a> <code>_lo</code>	Pointer to start of range.
<a href="#"></a> <code>_hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.690.2.9 do\_toupper()** [1/2]


```
template<typename _CharT>
virtual char\_type std::ctype<_CharT>::do_toupper (
    char\_type __c ) const    [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

<a href="#"></a> <code>_c</code>	The <code>char_type</code> to convert.
--	--

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::toupper()`.

**5.690.2.10 do\_toupper()** [2/2]

```
template<typename _CharT>
virtual const char\_type* std::ctype<_CharT>::do_toupper (
```



```
char_type * __lo,
const char_type * __hi ) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

<a href="#"> __lo</a>	Pointer to start of range.
<a href="#"> __hi</a>	Pointer to end of range.

#### Returns

[!\[\]\(73002692dd5e7a64e60946be3158e719\_img.jpg\) \\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 5.690.2.11 do\_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_widen (
    char __c ) const [protected], [virtual], [inherited]
```


Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<a href="#"> __c</a>	The char to convert.
---	----------------------

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::widen\(\)](#).

#### 5.690.2.12 do\_widen() [2/2]

```
template<typename _CharT>
virtual const char* std::ctype<\_CharT>::do\_widen (
    const char * __lo,
    const char * __hi,
    char\_type * __to ) const    [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the input to [char\\_type](#) using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

##### Parameters

<a href="#">_↵ _lo</a>	Pointer to start range.
<a href="#">_↵ _hi</a>	Pointer to end of range.
<a href="#">_↵ _to</a>	Pointer to the destination array.

##### Returns

[\\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 5.690.2.13 is() [1/2]

```
template<typename _CharT>
bool std::\_\_ctype\_abstract\_base<\_CharT>::is (
    mask __m,
    char\_type __c ) const    [inline], [inherited]
```

Test [char\\_type](#) classification.

This function finds a mask M for [\\_\\_c](#) and compares it to mask [\\_\\_m](#). It does so by returning the value of [ctype<char\\_↵  
type>::do\\_is\(\)](#).

## Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

## Returns

`(M & __m) != 0.`

Definition at line 169 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter>::get()`.

## 5.690.2.14 is() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::is (
    const char_type * __lo,
    const char_type * __hi,
    mask * __vec ) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

## Returns

`__hi.`

Definition at line 186 of file locale\_facets.h.

## 5.690.2.15 narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT>::narrow (
```



```
char_type __c,  
char __default ) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__c</code>	The char_type to convert.
<code>__default</code>	Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 331 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_InIter >::get(), and std::time\_put<\_CharT, \_OutIter >::put().

#### 5.690.2.16 narrow() [2/2]

```
template<typename _CharT>  
const char_type* std::__ctype_abstract_base<_CharT >::narrow (  
    const char_type * __lo,  
    const char_type * __hi,  
    char __default,  
    char * __to ) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, default is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**`__hi`.

Definition at line 353 of file locale\_facets.h.

**5.690.2.17 scan\_is()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_is (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

<code>↔ __m</code>	The mask to compare against.
<code>↔ __lo</code>	Pointer to start of range.
<code>↔ __hi</code>	Pointer to end of range.

**Returns**Pointer to matching char\_type if found, else `__hi`.

Definition at line 202 of file locale\_facets.h.




**5.690.2.18 scan\_not()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_not (
    mask __m,
    const char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

 _m	The mask to compare against.
 _lo	Pointer to first char in range.
 _hi	Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else \_\_hi.

Definition at line 218 of file locale\_facets.h.


**5.690.2.19 tolower()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
    char_type __c ) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

 _c	The char_type to convert.
---	---------------------------

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Definition at line 261 of file locale\_facets.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::get().

**5.690.2.20 tolower()** [2/2]



```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::tolower (
```

```
char_type * __lo,  
const char_type * __hi ) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

 <code>__lo</code>	Pointer to start of range.
 <code>__hi</code>	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 276 of file `locale_facets.h`.


**5.690.2.21 toupper()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
    char_type __c ) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

 <code>__c</code>	The <code>char_type</code> to convert.
--	--

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Definition at line 232 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _Inlter >::get()`.



**5.690.2.22 toupper()** [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::toupper (
    char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

<a href="#"></a> <code>__lo</code>	Pointer to start of range.
<a href="#"></a> <code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

Definition at line 247 of file locale\_facets.h.

## 5.690.2.23 widen() [1/2]


```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::widen (
    char __c ) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<a href="#"></a> <code>__c</code>	The char to convert.
---	----------------------

## Returns

The converted char\_type.

Definition at line 293 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::money_get<_CharT, _InIter>::do_get()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::money_put<_CharT, _OutIter>::do_put()`.

## 5.690.2.24 widen() [2/2]

```
template<typename _CharT>
const char* std::__ctype_abstract_base<_CharT>::widen (
```




```
const char * __lo,  
const char * __hi,  
char_type * __to ) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

<a href="#"> <code>__lo</code></a>	Pointer to start of range.
<a href="#"> <code>__hi</code></a>	Pointer to end of range.
<a href="#"> <code>__to</code></a>	Pointer to the destination array.

#### Returns

[`\_\_hi`](#).

Definition at line 312 of file locale\_facets.h.

### 5.690.3 Member Data Documentation

#### 5.690.3.1 id

```
template<typename _CharT>  
locale::id std::ctype<_CharT>::id [static], [inherited]
```

The facet id for ctype<char\_type>

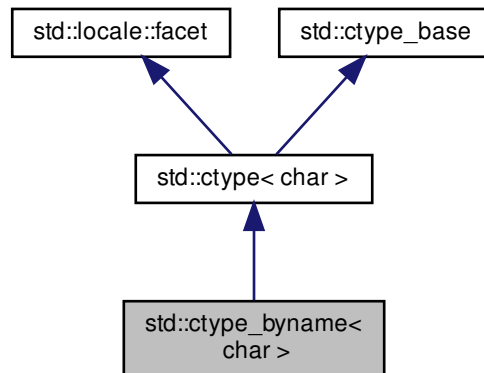
Definition at line 620 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.691 std::ctype\_byname&lt; char &gt; Class Template Reference

Inheritance diagram for std::ctype\_byname< char >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef char **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **ctype\_byname** (const string & \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char \_\_c) const
- const char \* **is** (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* **scan\_is** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* **scan\_not** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* **table** () const throw ()
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const



### Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault \_\_attribute\_\_((\_\_unused\_\_))) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault \_\_attribute\_\_((\_\_unused\_\_)), char \* \_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## Protected Attributes

- `__c_locale` **`_M_c_locale_ctype`**
- `bool` **`_M_del`**
- `char` **`_M_narrow`** `[1+static_cast< unsigned char >(-1)]`
- `char` **`_M_narrow_ok`**
- `const mask *` **`_M_table`**
- `__to_type` **`_M_tolower`**
- `__to_type` **`_M_toupper`**
- `char` **`_M_widen`** `[1+static_cast< unsigned char >(-1)]`
- `char` **`_M_widen_ok`**

## 5.691.1 Detailed Description

```
template<>
class std::ctype_byname< char >
```

22.2.1.4 Class `ctype_byname` specializations.

Definition at line 1499 of file `locale_facets.h`.

## 5.691.2 Member Typedef Documentation

5.691.2.1 `char_type`

```
typedef char std::ctype< char >::char_type [inherited]
```

Typedef for the template parameter `char`.

Definition at line 686 of file `locale_facets.h`.

## 5.691.3 Member Function Documentation

5.691.3.1 `classic_table()`

```
static const mask* std::ctype< char >::classic_table ( ) throw ( ) [static], [inherited]
```

Returns a pointer to the C locale mask table.

### 5.691.3.2 do\_narrow() [1/2]

```
virtual char std::ctype< char >::do_narrow (
    char_type __c,
    char __dfault __attribute__((unused)) ) const [inline], [protected], [virtual],
[inherited]
```

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__c</code>	The char to convert.
<code>__default</code>	Char to return if conversion fails.

## Returns

The converted char.

Definition at line 1134 of file locale\_facets.h.

## 5.691.3.3 do\_narrow() [2/2]

```
virtual const char_type* std::ctype< char >::do_narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __default __attribute__((unused)),
    char * __to ) const [inline], [protected], [virtual], [inherited]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 1160 of file locale\_facets.h.

**5.691.3.4 do\_tolower()** [1/2]


```
virtual char_type std::ctype< char >::do_tolower (
    char_type __c ) const    [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

 __c	The char to convert.
--	----------------------

**Returns**

The lowercase char if convertible, else \_\_c.

**5.691.3.5 do\_tolower()** [2/2]



```
virtual const char_type* std::ctype< char >::do_tolower (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

 __lo	Pointer to first char in range.
 __hi	Pointer to end of range.

**Returns**

\_\_hi.

## 5.691.3.6 do\_toupper() [1/2]


```
virtual char_type std::ctype< char >::do_toupper (
    char_type __c ) const    [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

## Parameters

 <code>__c</code>	The char to convert.
--	----------------------

## Returns

The uppercase char if convertible, else `__c`.

## 5.691.3.7 do\_toupper() [2/2]



```
virtual const char_type* std::ctype< char >::do_toupper (
    char_type * __lo,
    const char_type * __hi ) const    [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

## Parameters

 <code>__lo</code>	Pointer to start of range.
 <code>__hi</code>	Pointer to end of range.

## Returns

`__hi`.

**5.691.3.8 do\_widen()** [1/2]

```
virtual char_type std::ctype< char >::do_widen (  
    char __c ) const    [inline], [protected], [virtual], [inherited]
```

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

$\leftrightarrow$ __c	The char to convert.
--------------------------	----------------------

**Returns**

The converted character.

Definition at line 1084 of file locale\_facets.h.

**5.691.3.9 do\_widen()** [2/2]

```
virtual const char* std::ctype< char >::do_widen (  
    const char * __lo,  
    const char * __hi,  
    char_type * __to ) const    [inline], [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

$\leftrightarrow$ __lo	Pointer to start of range.
$\leftrightarrow$ __hi	Pointer to end of range.
$\leftrightarrow$ __to	Pointer to the destination array.

**Returns**`__hi`.

Definition at line 1107 of file locale\_facets.h.

**5.691.3.10 is()** [1/2]

```
bool std::ctype< char >::is (
    mask __m,
    char __c ) const [inline], [inherited]
```

Test char classification.

This function compares the mask table[c] to `__m`.**Parameters**

<code>__c</code>	The char to compare the mask of.
<code>__m</code>	The mask to compare against.

**Returns**True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file ctype\_inline.h.

**5.691.3.11 is()** [2/2]

```
const char * std::ctype< char >::is (
    const char * __lo,
    const char * __hi,
    mask * __vec ) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.



**Returns**

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

**5.691.3.12 narrow()** [1/2]

```
char std::ctype< char >::narrow (
    char_type __c,
    char __dfault ) const [inline], [inherited]
```

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

This function works as if it returns ctype<char>::do\_narrow(c). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

**Returns**

The converted character.

Definition at line 931 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_narrow()`.

**5.691.3.13 narrow()** [2/2]

```
const char_type* std::ctype< char >::narrow (
    const char_type * __lo,
    const char_type * __hi,
    char __dfault,
    char * __to ) const [inline], [inherited]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, default, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

**Returns**

`__hi`.

Definition at line 964 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_narrow()`.

**5.691.3.14 `scan_is()`**

```
const char * std::ctype< char >::scan_is (
    mask __m,
    const char * __lo,
    const char * __hi ) const [inline], [inherited]
```

Find char matching a mask.

This function searches for and returns the first char in `[lo,hi)` for which `is(m,char)` is true.

**Parameters**

<code>↵ __m</code>	The mask to compare against.
<code>↵ __lo</code>	Pointer to start of range.
<code>↵ __hi</code>	Pointer to end of range.

**Returns**

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file `ctype_inline.h`.




## 5.691.3.15 scan\_not()

```
const char * std::ctype< char >::scan_not (
    mask __m,
    const char * __lo,
    const char * __hi ) const [inline], [inherited]
```

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [\_\_lo,\_\_hi) for which is(m,char) is false.

## Parameters

 __m	The mask to compare against.
 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

## Returns

Pointer to a non-matching char if found, else \_\_hi.

Definition at line 67 of file ctype\_inline.h.

## 5.691.3.16 table()

```
const mask* std::ctype< char >::table ( ) const throw ( ) [inline], [inherited]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic\_table() if none was provided.

Definition at line 983 of file locale\_facets.h.

## 5.691.3.17 tolower() [1/2]


```
char_type std::ctype< char >::tolower (
    char_type __c ) const [inline], [inherited]
```

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_c). do\_tolower() must always return the same result for the same input.

**Parameters**

	The char to convert.
<code>__c</code>	

**Returns**

The lowercase char if convertible, else `__c`.

Definition at line 835 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_tolower()`.

**5.691.3.18 tolower()** [2/2]



```
const char_type* std::ctype< char >::tolower (
    char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

**Parameters**

	Pointer to first char in range.
<code>__lo</code>	
	Pointer to end of range.
<code>__hi</code>	

**Returns**

`__hi`.

Definition at line 852 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_tolower()`.

## 5.691.3.19 toupper() [1/2]

```
char_type std::ctype< char >::toupper (
    char_type __c ) const [inline], [inherited]
```

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do\_toupper(c). do\_toupper() must always return the same result for the same input.

**Parameters**

$\leftrightarrow$ __c	The char to convert.
--------------------------	----------------------

**Returns**

The uppercase char if convertible, else \_\_c.

Definition at line 802 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

## 5.691.3.20 toupper() [2/2]

```
const char_type* std::ctype< char >::toupper (
    char_type * __lo,
    const char_type * __hi ) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char in the range [\_\_lo,\_\_hi) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>:: do\_toupper(\_\_lo, \_\_hi). do\_toupper() must always return the same result for the same input.

**Parameters**

$\leftrightarrow$ __lo	Pointer to first char in range.
$\leftrightarrow$ __hi	Pointer to end of range.

**Returns**

`__hi`.

Definition at line 819 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_toupper()`.

**5.691.3.21 widen()** [1/2]

```
char_type std::ctype< char >::widen (  
    char __c ) const    [inline], [inherited]
```

Widen char.

This function converts the `char` to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

<code>__c</code>	The char to convert.
------------------	----------------------

**Returns**

The converted character.

Definition at line 872 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

**5.691.3.22 widen()** [2/2]

```
const char* std::ctype< char >::widen (  
    const char * __lo,  
    const char * __hi,  
    char_type * __to ) const    [inline], [inherited]
```

Widen char array.

This function converts each `char` in the input to `char` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

<code>_↔ _lo</code>	Pointer to first char in range.
<code>_↔ _hi</code>	Pointer to end of range.
<code>_↔ _to</code>	Pointer to the destination array.

## Returns

`__hi`.

Definition at line 899 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

## 5.691.4 Member Data Documentation

5.691.4.1 `id`

```
locale::id std::ctype< char >::id [static], [inherited]
```

The facet id for `ctype<char>`

Definition at line 703 of file `locale_facets.h`.

5.691.4.2 `table_size`

```
const size_t std::ctype< char >::table_size [static], [inherited]
```

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 705 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

5.692 `std::decay<_Tp>` Class Template Reference

## Public Types

- `typedef __decay_selector< __remove_type >::__type type`



### 5.692.1 Detailed Description

```
template<typename _Tp>
class std::decay< _Tp >
```

decay

Definition at line 1871 of file type\_traits.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 5.693 std::decimal::decimal128 Class Reference

### Public Types

- typedef float \_\_decfloat128 **\_\_attribute\_\_**((mode(TD)))

### Public Member Functions

- **decimal128** ([decimal32](#) d32)
- **decimal128** ([decimal64](#) d64)
- **decimal128** (float \_\_r)
- **decimal128** (double \_\_r)
- **decimal128** (long double \_\_r)
- **decimal128** (int \_\_z)
- **decimal128** (unsigned int \_\_z)
- **decimal128** (long \_\_z)
- **decimal128** (unsigned long \_\_z)
- **decimal128** (long long \_\_z)
- **decimal128** (unsigned long long \_\_z)
- [decimal128](#) (\_\_decfloat128 \_\_z)
- \_\_decfloat128 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat128 \_\_x)
- **operator long long** () const
- [decimal128](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator\*=** (int \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator\*=** (long \_\_rhs)
- [decimal128](#) & **operator\*=** (long long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=** ([decimal32](#) \_\_rhs)

- [decimal128](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator+=** (long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator+=** (long long \_\_rhs)
- [decimal128](#) & **operator+=** (int \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=** (int \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator-=** (long long \_\_rhs)
- [decimal128](#) & **operator-=** (long \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** (long \_\_rhs)
- [decimal128](#) & **operator/=** (long long \_\_rhs)
- [decimal128](#) & **operator/=** (int \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) \_\_rhs)

#### 5.693.1 Detailed Description

##### 3.2.4 Class decimal128.

Definition at line 399 of file decimal.

#### 5.693.2 Constructor & Destructor Documentation

##### 5.693.2.1 decimal128()

```
std::decimal::decimal128::decimal128 (
    __decfloat128 __z ) [inline]
```

Conforming extension: Conversion from scalar decimal type.

Definition at line 424 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.694 std::decimal::decimal32 Class Reference

### Public Types

- typedef float \_\_decfloat32 \_\_attribute\_\_((mode(SD)))

### Public Member Functions

- **decimal32** ([decimal64](#) \_\_d64)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** (float \_\_r)
- **decimal32** (double \_\_r)
- **decimal32** (long double \_\_r)
- **decimal32** (int \_\_z)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (long long \_\_z)
- **decimal32** (unsigned long long \_\_z)
- [decimal32](#) (\_\_decfloat32 \_\_z)
- \_\_decfloat32 \_\_getval (void)
- void \_\_setval (\_\_decfloat32 \_\_x)
- **operator long long** () const
- [decimal32](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator\*=** (int \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator\*=** (long \_\_rhs)
- [decimal32](#) & **operator\*=** (long long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator+=** (long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator+=** (long long \_\_rhs)
- [decimal32](#) & **operator+=** (int \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** (int \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator-=** (long long \_\_rhs)

- [decimal32](#) & **operator=** (long \_\_rhs)
- [decimal32](#) & **operator=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** (long \_\_rhs)
- [decimal32](#) & **operator/=** (long long \_\_rhs)
- [decimal32](#) & **operator/=** (int \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) \_\_rhs)

#### 5.694.1 Detailed Description

##### 3.2.2 Class decimal32.

Definition at line 227 of file decimal.

#### 5.694.2 Constructor & Destructor Documentation

##### 5.694.2.1 decimal32()

```
std::decimal::decimal32::decimal32 (
    __decfloat32 __z ) [inline]
```

Conforming extension: Conversion from scalar decimal type.

Definition at line 251 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.695 std::decimal::decimal64 Class Reference

### Public Types

- typedef float \_\_decfloat64 **\_\_attribute\_\_**((mode(DD)))

## Public Member Functions

- **decimal64** ([decimal32](#) d32)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (float \_\_r)
- **decimal64** (double \_\_r)
- **decimal64** (long double \_\_r)
- **decimal64** (int \_\_z)
- **decimal64** (unsigned int \_\_z)
- **decimal64** (long \_\_z)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (long long \_\_z)
- **decimal64** (unsigned long long \_\_z)
- [decimal64](#) (\_\_decfloat64 \_\_z)
- \_\_decfloat64 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- **operator long long** () const
- [decimal64](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator\*=** (int \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator\*=** (long \_\_rhs)
- [decimal64](#) & **operator\*=** (long long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator+=** (long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator+=** (long long \_\_rhs)
- [decimal64](#) & **operator+=** (int \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=** (int \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator-=** (long long \_\_rhs)
- [decimal64](#) & **operator-=** (long \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator/=** (unsigned long long \_\_rhs)

- [decimal64](#) & `operator/=` (long \_\_rhs)
- [decimal64](#) & `operator/=` (long long \_\_rhs)
- [decimal64](#) & `operator/=` (int \_\_rhs)
- [decimal64](#) & `operator/=` ([decimal128](#) \_\_rhs)
- [decimal64](#) & `operator/=` (unsigned int \_\_rhs)
- [decimal64](#) & `operator/=` ([decimal32](#) \_\_rhs)

#### 5.695.1 Detailed Description

##### 3.2.3 Class decimal64.

Definition at line 313 of file decimal.

#### 5.695.2 Constructor & Destructor Documentation

##### 5.695.2.1 decimal64()

```
std::decimal::decimal64::decimal64 (
    __decfloat64 __z ) [inline]
```

Conforming extension: Conversion from scalar decimal type.

Definition at line 337 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.696 `std::default_delete<_Tp>` Struct Template Reference

### Public Member Functions

- constexpr [default\\_delete](#) () noexcept=default
- template<typename \_Up, typename = typename enable\_if<is\_convertible<\_Up\*, \_Tp\*>::value>::type> [default\\_delete](#) (const [default\\_delete](#)<\_Up> &) noexcept
- void [operator\(\)](#) (\_Tp \*\_\_ptr) const

#### 5.696.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete<_Tp>
```

Primary template of default\_delete, used by unique\_ptr.

Definition at line 59 of file unique\_ptr.h.

## 5.696.2 Constructor & Destructor Documentation

### 5.696.2.1 `default_delete()` [1/2]

```
template<typename _Tp >  
constexpr std::default\_delete< _Tp >::default\_delete ( ) [default], [noexcept]
```

Default constructor.

### 5.696.2.2 `default_delete()` [2/2]

```
template<typename _Tp >  
template<typename _Up , typename = typename enable_if<is_convertible<_Up*, _Tp*>::value>::type>  
std::default\_delete< _Tp >::default\_delete (   
    const default\_delete< _Up > & ) [inline], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for arrays of another type, `_Up`, only if `_Up*` is convertible to `_Tp*`.

Definition at line 71 of file `unique_ptr.h`.

## 5.696.3 Member Function Documentation

### 5.696.3.1 `operator>()`

```
template<typename _Tp >  
void std::default\_delete< _Tp >::operator() (   
    _Tp * __ptr ) const [inline]
```

Calls `delete __ptr`.

Definition at line 75 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.697 std::default\_delete&lt; \_Tp[]&gt; Struct Template Reference

## Public Member Functions

- constexpr [default\\_delete](#) () noexcept=default
- template<typename \_Up , typename = typename enable\_if<is\_convertible<\_Up(\*)[], \_Tp(\*)[]>::value>::type> [default\\_delete](#) (const [default\\_delete](#)< \_Up[]> &) noexcept
- template<typename \_Up > [enable\\_if](#)< [is\\_convertible](#)< \_Up(\*)[], \_Tp(\*)[]>::value >::type [operator\(\)](#) (\_Up \* \_\_ptr) const

## 5.697.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete< _Tp[]>
```

Specialization for arrays, [default\\_delete](#).

Definition at line 89 of file [unique\\_ptr.h](#).

## 5.697.2 Constructor &amp; Destructor Documentation

5.697.2.1 [default\\_delete\(\)](#) [1/2]

```
template<typename _Tp >
constexpr std::default\_delete< _Tp[]>::default\_delete ( ) [default], [noexcept]
```

Default constructor.

5.697.2.2 [default\\_delete\(\)](#) [2/2]

```
template<typename _Tp >
template<typename _Up , typename = typename enable_if<is_convertible<_Up(*)[], _Tp(*)[]>::value>↵
::type>
std::default\_delete< _Tp[]>::default\_delete (
    const default\_delete< _Up[]> & ) [inline], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for arrays of another type, such as a const-qualified version of [\\_Tp](#).

Conversions from types derived from [\\_Tp](#) are not allowed because it is unsafe to [delete\[\]](#) an array of derived types through a pointer to the base type.

Definition at line 106 of file [unique\\_ptr.h](#).



### 5.697.3 Member Function Documentation

#### 5.697.3.1 operator()

```
template<typename _Tp >
template<typename _Up >
enable_if<is_convertible<_Up(*)[], _Tp(*)[]>::value>::type std::default_delete<_Tp[]>::operator()
(
    _Up * __ptr ) const [inline]
```

Calls `delete[] __ptr`.

Definition at line 111 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

### 5.698 std::defer\_lock\_t Struct Reference

#### 5.698.1 Detailed Description

Do not acquire ownership of the mutex.

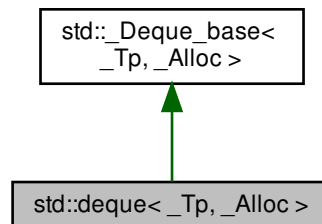
Definition at line 132 of file `std_mutex.h`.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

### 5.699 std::deque<\_Tp, \_Alloc> Class Template Reference

Inheritance diagram for `std::deque<_Tp, _Alloc>`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `deque` ()
- `deque` (const allocator\_type &\_\_a)
- `deque` (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- `deque` (size\_type \_\_n, const value\_type &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- `deque` (const deque &\_\_x)
- `deque` (deque &&\_\_x)
- `deque` (const deque &\_\_x, const allocator\_type &\_\_a)
- `deque` (deque &&\_\_x, const allocator\_type &\_\_a)
- `deque` (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  `deque` (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- `~deque` ()
- void `assign` (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (initializer\_list< value\_type > \_\_l)
- reference `at` (size\_type \_\_n)
- const\_reference `at` (size\_type \_\_n) const
- reference `back` () noexcept
- const\_reference `back` () const noexcept
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- void `clear` () noexcept
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- template<typename... \_Args>  
  `iterator emplace` (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
  void `emplace_back` (\_Args &&... \_\_args)
- template<typename... \_Args>  
  void `emplace_front` (\_Args &&... \_\_args)

- bool `empty` () const noexcept
- iterator `end` () noexcept
- const\_iterator `end` () const noexcept
- iterator `erase` (const\_iterator \_\_position)
- iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- reference `front` () noexcept
- const\_reference `front` () const noexcept
- allocator\_type `get_allocator` () const noexcept
- iterator `insert` (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
- iterator `insert` (const\_iterator \_\_p, initializer\_list< value\_type > \_\_l)
- iterator `insert` (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
iterator `insert` (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type `max_size` () const noexcept
- deque & `operator=` (const deque &\_\_x)
- deque & `operator=` (deque &&\_\_x) noexcept(\_Alloc\_traits::\_S\_always\_equal())
- deque & `operator=` (initializer\_list< value\_type > \_\_l)
- reference `operator[]` (size\_type \_\_n) noexcept
- const\_reference `operator[]` (size\_type \_\_n) const noexcept
- void `pop_back` () noexcept
- void `pop_front` () noexcept
- void `push_back` (const value\_type &\_\_x)
- void `push_back` (value\_type &&\_\_x)
- void `push_front` (const value\_type &\_\_x)
- void `push_front` (value\_type &&\_\_x)
- reverse\_iterator `rbegin` () noexcept
- const\_reverse\_iterator `rbegin` () const noexcept
- reverse\_iterator `rend` () noexcept
- const\_reverse\_iterator `rend` () const noexcept
- void `resize` (size\_type \_\_new\_size)
- void `resize` (size\_type \_\_new\_size, const value\_type &\_\_x)
- void `shrink_to_fit` () noexcept
- size\_type `size` () const noexcept
- void `swap` (deque &\_\_x) noexcept

## Protected Types

- enum { `_S_initial_map_size` }
- typedef `__gnu_cxx::__alloc_traits`< \_Map\_alloc\_type > `_Map_alloc_traits`
- typedef \_Alloc\_traits::template rebind< \_Ptr >::other `_Map_alloc_type`
- typedef \_Alloc\_traits::pointer `_Ptr`
- typedef \_Alloc\_traits::const\_pointer `_Ptr_const`

## Protected Member Functions

- `_Map_pointer _M_allocate_map (size_t __n)`
- `_Map_pointer _M_allocate_map (size_t __n)`
- `_Ptr _M_allocate_node ()`
- `_Ptr _M_allocate_node ()`
- `template<typename _InputIterator >`  
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _Integer >`  
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Map_pointer __nstart, _Map_pointer __nfinish)`
- `void _M_create_nodes (_Map_pointer __nstart, _Map_pointer __nfinish)`
- `void _M_deallocate_map (_Map_pointer __p, size_t __n) noexcept`
- `void _M_deallocate_map (_Map_pointer __p, size_t __n) noexcept`
- `void _M_deallocate_node (_Ptr __p) noexcept`
- `void _M_deallocate_node (_Ptr __p) noexcept`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`  
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator< _Tp > &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Map_pointer __nstart, _Map_pointer __nfinish) noexcept`
- `void _M_destroy_nodes (_Map_pointer __nstart, _Map_pointer __nfinish) noexcept`
- `iterator _M_erase (iterator __pos)`
- `iterator _M_erase (iterator __first, iterator __last)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void \_M\_fill\_initialize (const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Map_alloc_type _M_get_map_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void \_M\_initialize\_map (size_t)`
- `void \_M\_initialize\_map (size_t)`
- `template<typename... _Args>`  
`iterator \_M\_insert\_aux (iterator __pos, _Args &&... __args)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type &__x)`
- `template<typename _ForwardIterator >`  
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`

- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
  - `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
  - `void _M_move_assign1 (deque &&__x, true_type) noexcept`
  - `void _M_move_assign1 (deque &&__x, false_type)`
  - `void _M_move_assign2 (deque &&__x, true_type)`
  - `void _M_move_assign2 (deque &&__x, false_type)`
  - `void _M_range_check (size_type __n) const`
  - `template<typename _InputIterator >`  
`void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
  - `template<typename _ForwardIterator >`  
`void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
  - `template<typename... _Args>`  
`void _M_replace_map (_Args &&... __args)`
  - `bool _M_shrink_to_fit ()`
- 
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
  - `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- 
- `template<typename... _Args>`  
`void _M_push_back_aux (_Args &&... __args)`
  - `template<typename... _Args>`  
`void _M_push_front_aux (_Args &&... __args)`
  - `void _M_pop_back_aux ()`
  - `void _M_pop_front_aux ()`
- 
- `iterator _M_reserve_elements_at_front (size_type __n)`
  - `iterator _M_reserve_elements_at_back (size_type __n)`
  - `void _M_new_elements_at_front (size_type __new_elements)`
  - `void _M_new_elements_at_back (size_type __new_elements)`
- 
- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
  - `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
  - `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

## Static Protected Member Functions

- static size\_t **S\_buffer\_size** () noexcept

## Protected Attributes

- \_Deque\_impl [\\_M\\_impl](#)

## 5.699.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::deque< _Tp, _Alloc >
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

`map_size` is at least 8. `map` is an array of `map_size` pointers-to-*nodes*. (The name `map` has nothing to do with the `std::map` class, and **nodes** should not be confused with `std::list`'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-`Tp`. If `Tp` is very large, there will be one `Tp` element per node (i.e., an *array* of one). For non-huge `Tp`'s, node size is inversely related to `Tp` size: the larger the `Tp`, the fewer `Tp`'s will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different `Tp`'s, to improve allocator efficiency.

Not every pointer in the `map` array will point to a node. If the initial number of elements in the deque is small, the /middle/ `map` pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the `map` grows: available `map` pointers, if any, will be on the ends. As new nodes are created, only a subset of the `map`'s pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator *i*:
  - *i*.node points to a member of the map array. (Yes, you read that correctly: *i*.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - *i*.first == \*(*i*.node) (This points to the node (first *Tp* element).)
  - *i*.last == *i*.first + node\_size
  - *i*.cur is a pointer in the range [*i*.first, *i*.last). NOTE: the implication of this is that *i*.cur is always a dereferenceable pointer, even if *i* is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.↔node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.↔first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map\_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map\_size).
- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, `_Base`, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 832 of file `stl_deque.h`.

## 5.699.2 Constructor & Destructor Documentation

### 5.699.2.1 deque() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque ( ) [inline]
```

Creates a deque with no elements.

Definition at line 898 of file `stl_deque.h`.

### 5.699.2.2 deque() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    const allocator_type & __a ) [inline], [explicit]
```

Creates a deque with no elements.

## Parameters

$\_a$	An allocator object.
-------	----------------------

Definition at line 905 of file stl\_deque.h.

## 5.699.2.3 deque() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    size_type __n,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a deque with default constructed elements.

## Parameters

$\_n$	The number of elements to initially create.
$\_a$	An allocator.

This constructor fills the deque with  $n$  default constructed elements.

Definition at line 918 of file stl\_deque.h.

## 5.699.2.4 deque() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type() ) [inline]
```

Creates a deque with copies of an exemplar element.

## Parameters

$\_n$	The number of elements to initially create.
$\_value$	An element to copy.
$\_a$	An allocator.

This constructor fills the deque with  $\_n$  copies of  $\_value$ .



Definition at line 930 of file `stl_deque.h`.

#### 5.699.2.5 `deque()` [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    const deque< _Tp, _Alloc > & __x ) [inline]
```

Deque copy constructor.

##### Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 957 of file `stl_deque.h`.

#### 5.699.2.6 `deque()` [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    deque< _Tp, _Alloc > && __x ) [inline]
```

Deque move constructor.

##### Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The newly-created deque contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified deque.

Definition at line 972 of file `stl_deque.h`.

#### 5.699.2.7 `deque()` [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    const deque< _Tp, _Alloc > & __x,
    const allocator_type & __a ) [inline]
```

Copy constructor with alternative allocator.

Definition at line 976 of file stl\_deque.h.

#### 5.699.2.8 deque() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    deque< _Tp, _Alloc > && __x,
    const allocator_type & __a ) [inline]
```

Move constructor with alternative allocator.

Definition at line 983 of file stl\_deque.h.

#### 5.699.2.9 deque() [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a deque from an initializer list.

##### Parameters

$\_l$	An initializer_list.
$\_a$	An allocator object.

Create a deque consisting of copies of the elements in the initializer\_list  $\_l$ .

This will call the element type's copy constructor N times (where N is  $\_l.size()$ ) and do no memory reallocation.

Definition at line 1006 of file stl\_deque.h.

#### 5.699.2.10 deque() [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
std::deque< _Tp, _Alloc >::deque (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a deque from a range.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator object.

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor *N* times (where *N* is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most *2N* calls to the copy constructor, and *logN* memory reallocations.

Definition at line 1033 of file `stl_deque.h`.

**5.699.2.11 ~deque()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::~~deque ( ) [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1054 of file `stl_deque.h`.

**5.699.3 Member Function Documentation****5.699.3.1 \_M\_fill\_initialize()**

```
template<typename _Tp , typename _Alloc >
void deque::_M_fill_initialize (
    const value_type & __value ) [protected]
```

Fills the deque with copies of `value`.

**Parameters**

<code>__value</code>	Initial value.
----------------------	----------------

**Returns**

Nothing.

**Precondition**

\_M\_start and \_M\_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 392 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::deque().

**5.699.3.2 \_M\_initialize\_map() [1/2]**

```
template<typename _Tp , typename _Alloc >
void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (
    size_t __num_elements ) [protected], [inherited]
```

Layout storage.

**Parameters**

<code>__num_elements</code>	The count of T's for which to allocate space at first.
-----------------------------	--

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 683 of file stl\_deque.h.

**5.699.3.3 \_M\_initialize\_map() [2/2]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map [protected]
```

Layout storage.

**Parameters**

<code>__num_elements</code>	The count of T's for which to allocate space at first.
-----------------------------	--

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 683 of file `stl_deque.h`.

#### 5.699.3.4 `_M_new_elements_at_back()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_new_elements_at_back (
    size_type __new_elements ) [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 894 of file `deque.tcc`.

Referenced by `std::deque<_StateSeqT >::_M_reserve_elements_at_back()`.

#### 5.699.3.5 `_M_new_elements_at_front()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_new_elements_at_front (
    size_type __new_elements ) [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 869 of file `deque.tcc`.

Referenced by `std::deque<_StateSeqT >::_M_reserve_elements_at_front()`.

#### 5.699.3.6 `_M_pop_back_aux()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_pop_back_aux ( ) [protected]
```

Helper functions for `push_*` and `pop_*`.

Definition at line 548 of file `deque.tcc`.

#### 5.699.3.7 `_M_pop_front_aux()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_pop_front_aux ( ) [protected]
```

Helper functions for `push_*` and `pop_*`.

Definition at line 564 of file `deque.tcc`.

**5.699.3.8** \_M\_push\_back\_aux()

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
void deque::_M_push_back_aux (
    _Args &&... __args ) [protected]
```

Helper functions for push\_\* and pop\_\*.

Definition at line 480 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::push\_back().

**5.699.3.9** \_M\_push\_front\_aux()

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
void deque::_M_push_front_aux (
    _Args &&... __args ) [protected]
```

Helper functions for push\_\* and pop\_\*.

Definition at line 515 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::push\_front().

**5.699.3.10** \_M\_range\_check()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_range_check (
    size_type __n ) const [inline], [protected]
```

Safety check used only from at().

Definition at line 1410 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::at().

**5.699.3.11** \_M\_range\_initialize() [1/2]

```
template<typename _Tp , typename _Alloc >
template<typename _InputIterator >
void deque::_M_range_initialize (
    _InputIterator __first,
    _InputIterator __last,
    std::input_iterator_tag ) [protected]
```

Fills the deque with whatever is in [first,last).

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 418 of file `deque.tcc`.

Referenced by `std::deque<_StateSeqT>::deque()`.

**5.699.3.12 `_M_range_initialize()` [2/2]**

```
template<typename _Tp , typename _Alloc >
template<typename _ForwardIterator >
void deque::_M_range_initialize (
    _ForwardIterator __first,
    _ForwardIterator __last,
    std::forward_iterator_tag ) [protected]
```

Fills the deque with whatever is in `[first,last)`.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 442 of file `deque.tcc`.

**5.699.3.13 \_M\_reallocate\_map()**

```
template<typename _Tp, typename _Alloc >
void deque::_M_reallocate_map (
    size_type __nodes_to_add,
    bool __add_at_front ) [protected]
```

Memory-handling helpers for the major map.

Makes sure the \_M\_map has space for new nodes. Does not actually add the nodes. Can invalidate \_M\_map pointers. (And consequently, deque iterators.)

Definition at line 919 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::\_M\_reserve\_map\_at\_back(), and std::deque< \_StateSeqT >::\_M\_reserve\_map\_at\_front().

**5.699.3.14 \_M\_reserve\_elements\_at\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back (
    size_type __n ) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 2133 of file stl\_deque.h.

**5.699.3.15 \_M\_reserve\_elements\_at\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front (
    size_type __n ) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 2123 of file stl\_deque.h.

**5.699.3.16 \_M\_reserve\_map\_at\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (
    size_type __nodes_to_add = 1 ) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the \_M\_map has space for new nodes. Does not actually add the nodes. Can invalidate \_M\_map pointers. (And consequently, deque iterators.)

Definition at line 2159 of file stl\_deque.h.



**5.699.3.17** `_M_reserve_map_at_front()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (
    size_type __nodes_to_add = 1 ) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 2167 of file `stl_deque.h`.

**5.699.3.18** `assign()` [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
    size_type __n,
    const value_type & __val ) [inline]
```

Assigns a given value to a deque.

**Parameters**

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a deque with  $n$  copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

Definition at line 1117 of file `stl_deque.h`.

**5.699.3.19** `assign()` [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
void std::deque< _Tp, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Assigns a range to a deque.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a deque with copies of the elements in the range [\_\_first,\_\_last).

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.






Definition at line 1136 of file stl\_deque.h.

#### 5.699.3.20 assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
    initializer_list< value_type > __l ) [inline]
```

Assigns an initializer list to a deque.

##### Parameters

	An initializer_list.
	
	
	
	

This function fills a deque with copies of the elements in the initializer\_list \_\_l.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.



Definition at line 1161 of file stl\_deque.h.

#### 5.699.3.21 at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::at (
    size_type __n ) [inline]
```

Provides access to the data contained in the deque.

##### Parameters

 	The index of the element for which data should be accessed.
--	---

**Returns**

Read/write reference to data.

**Exceptions**

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1432 of file `stl_deque.h`.

**5.699.3.22 at()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::at (
    size_type __n ) const [inline]
```

Provides access to the data contained in the deque.

**Parameters**

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

**Returns**

Read-only (constant) reference to data.

**Exceptions**

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1450 of file `stl_deque.h`.

**5.699.3.23 back()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::back ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1483 of file stl\_deque.h.

#### 5.699.3.24 back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::back ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1496 of file stl\_deque.h.

#### 5.699.3.25 begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1176 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::clear(), std::deque< \_StateSeqT >::deque(), std::deque< \_StateSeqT >::insert(), std::operator<(), std::deque< \_StateSeqT >::operator=(), std::operator==( ), and std::deque< \_StateSeqT >::~~deque().

#### 5.699.3.26 begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1184 of file stl\_deque.h.

#### 5.699.3.27 cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1247 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::insert().

**5.699.3.28 cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1256 of file `stl_deque.h`.

**5.699.3.29 clear()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1850 of file `stl_deque.h`.

**5.699.3.30 crbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1265 of file `stl_deque.h`.

**5.699.3.31 crend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1274 of file `stl_deque.h`.

**5.699.3.32 emplace()**

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
deque< _Tp, _Alloc >::iterator deque::emplace (
    const_iterator __position,
    _Args &&... __args )
```

Inserts an object in deque before specified iterator.

## Parameters

<code>__position</code>	A const_iterator into the deque.
<code>__args</code>	Arguments.

## Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location.

Definition at line 186 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::insert().

## 5.699.3.33 empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the deque is empty. (Thus begin() would equal end().)

Definition at line 1367 of file stl\_deque.h.

## 5.699.3.34 end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1193 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::deque(), std::operator<(), std::deque< \_StateSeqT >::operator=(), std::deque< \_StateSeqT >::operator==(), and std::deque< \_StateSeqT >::~~deque().

## 5.699.3.35 end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1202 of file stl\_deque.h.

## 5.699.3.36 erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
    const_iterator __position ) [inline]
```

Remove element at given position.

**Parameters**

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1790 of file `stl_deque.h`.

**5.699.3.37 erase()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Remove a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [`__first`,`__last`) and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1814 of file `stl_deque.h`.

**5.699.3.38 front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::front ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1461 of file `stl_deque.h`.

**5.699.3.39** front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::front ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1472 of file stl\_deque.h.

**5.699.3.40** get\_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 1167 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::operator=().

**5.699.3.41** insert() [1/5]

```
template<typename _Tp , typename _Alloc >
deque< _Tp, _Alloc >::iterator deque::insert (
    const_iterator __position,
    const value_type & __x )
```

Inserts given value into deque before specified iterator.

**Parameters**

<code>__position</code>	A const_iterator into the deque.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 210 of file deque.tcc.



**5.699.3.42** `insert()` [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __position,
    value_type && __x ) [inline]
```

Inserts given rvalue into deque before specified iterator.

**Parameters**

<code>__position</code>	A const_iterator into the deque.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1675 of file `stl_deque.h`.

**5.699.3.43** `insert()` [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __p,
    initializer_list< value_type > __l ) [inline]
```

Inserts an initializer list into the deque.

**Parameters**

<code>__p</code>	An iterator into the deque.
<code>__l</code>	An initializer_list.

This function will insert copies of the data in the initializer\_list `__l` into the deque before the location specified by `__p`. This is known as *list insert*.

Definition at line 1688 of file `stl_deque.h`.

## 5.699.3.44 insert() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x ) [inline]
```

Inserts a number of copies of given data into the deque.

## Parameters

<code>__position</code>	A const_iterator into the deque.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

## Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

Definition at line 1709 of file `stl_deque.h`.

## 5.699.3.45 insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
iterator std::deque< _Tp, _Alloc >::insert (
    const_iterator __position,
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Inserts a range into the deque.

## Parameters

<code>__position</code>	A const_iterator into the deque.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

## Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

Definition at line 1745 of file `stl_deque.h`.

#### 5.699.3.46 `max_size()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the size() of the largest possible deque.

Definition at line 1286 of file `stl_deque.h`.

#### 5.699.3.47 `operator=()` [1/3]

```
template<typename _Tp , typename _Alloc >
deque< _Tp, _Alloc > & deque::operator= (
    const deque< _Tp, _Alloc > & __x )
```

Deque assignment operator.

##### Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

All the elements of `x` are copied.

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 94 of file `deque.tcc`.

#### 5.699.3.48 `operator=()` [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (
    deque< _Tp, _Alloc > && __x ) [inline], [noexcept]
```

Deque move assignment operator.

##### Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The contents of `__x` are moved into this deque (without copying, if the allocators permit it). `__x` is a valid, but unspecified deque.






Definition at line 1079 of file `stl_deque.h`.

#### 5.699.3.49 operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Assigns an initializer list to a deque.

##### Parameters

	An initializer_list.
	
	
	
	

This function fills a deque with copies of the elements in the initializer\_list `__l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.



Definition at line 1098 of file `stl_deque.h`.

#### 5.699.3.50 operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::operator[] (
    size_type __n ) [inline], [noexcept]
```

Subscript access to the data contained in the deque.

##### Parameters

	The index of the element for which data should be accessed.
	

##### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1383 of file `stl_deque.h`.

#### 5.699.3.51 `operator[]()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::operator[] (
    size_type __n ) const [inline], [noexcept]
```

Subscript access to the data contained in the deque.

##### Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

##### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1401 of file `stl_deque.h`.

#### 5.699.3.52 `pop_back()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_back ( ) [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1611 of file `stl_deque.h`.

#### 5.699.3.53 `pop_front()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front ( ) [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1588 of file `stl_deque.h`.

### 5.699.3.54 push\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (
    const value_type & __x ) [inline]
```

Add data to the end of the deque.

#### Parameters

$\_x$	Data to be added.
-------	-------------------

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1552 of file stl\_deque.h.

### 5.699.3.55 push\_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (
    const value_type & __x ) [inline]
```

Add data to the front of the deque.

#### Parameters

$\_x$	Data to be added.
-------	-------------------

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1515 of file stl\_deque.h.

### 5.699.3.56 rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1211 of file stl\_deque.h.

**5.699.3.57** `rbegin()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1220 of file `stl_deque.h`.

**5.699.3.58** `rend()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1229 of file `stl_deque.h`.

**5.699.3.59** `rend()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1238 of file `stl_deque.h`.

**5.699.3.60** `resize()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
    size_type __new_size ) [inline]
```

Resizes the deque to the specified number of elements.

**Parameters**

<code>__new_size</code>	Number of elements the deque should contain.
-------------------------	--

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1300 of file stl\_deque.h.

#### 5.699.3.61 resize() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
    size_type __new_size,
    const value_type & __x ) [inline]
```

Resizes the deque to the specified number of elements.

##### Parameters

<code>__new_size</code>	Number of elements the deque should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1322 of file stl\_deque.h.

#### 5.699.3.62 shrink\_to\_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline], [noexcept]
```

A non-binding request to reduce memory use.

Definition at line 1358 of file stl\_deque.h.

#### 5.699.3.63 size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the number of elements in the deque.

Definition at line 1281 of file stl\_deque.h.

Referenced by `std::deque< _StateSeqT >::_M_range_check()`, `std::deque< _StateSeqT >::operator=()`, `std::deque< _StateSeqT >::operator==()`, and `std::deque< _StateSeqT >::resize()`.

#### 5.699.3.64 swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::swap (
    deque< _Tp, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another deque.



## Parameters

<a href="#"><code>_↔</code></a>	A deque of the same element and allocator types.
<a href="#"><code>_X</code></a>	

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1832 of file `stl_deque.h`.

## 5.699.4 Member Data Documentation

5.699.4.1 `_M_impl`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Deque_impl std::_Deque_base< _Tp, _Alloc >::_M_impl [protected]
```

A total of four data members accumulated down the hierarchy. May be accessed via `_M_impl.*`

Definition at line 634 of file `stl_deque.h`.

The documentation for this class was generated from the following files:

- [stl\\_deque.h](#)
- [deque.tcc](#)

5.700 `std::discard_block_engine< _RandomNumberEngine, __p, __r >` Class Template Reference

## Public Types

- typedef `_RandomNumberEngine::result_type` [result\\_type](#)

## Public Member Functions

- [discard\\_block\\_engine](#) ()
- [discard\\_block\\_engine](#) (const `_RandomNumberEngine` &\_\_rng)
- [discard\\_block\\_engine](#) (`_RandomNumberEngine` &&\_\_rng)
- [discard\\_block\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_↔ same<_Sseq, _RandomNumberEngine>::value> ::type>`  
[discard\\_block\\_engine](#) (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- void [seed](#) ()
- void [seed](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` >  
void [seed](#) (`_Sseq` &\_\_q)

### Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

### Static Public Attributes

- static constexpr size\_t **block\_size**
- static constexpr size\_t **used\_block**

### Friends

- template<typename \_RandomNumberEngine1, size\_t \_\_p1, size\_t \_\_r1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::discard\\_block\\_engine](#)< \_RandomNumberEngine1, \_\_p1, \_\_r1 > &\_\_x)
- bool [operator==](#) (const [discard\\_block\\_engine](#) &\_\_lhs, const [discard\\_block\\_engine](#) &\_\_rhs)
- template<typename \_RandomNumberEngine1, size\_t \_\_p1, size\_t \_\_r1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::discard\\_block\\_engine](#)< \_RandomNumberEngine1, \_\_p1, \_\_r1 > &\_\_x)

#### 5.700.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine<_RandomNumberEngine, __p, __r>
```

Produces random numbers from some base engine by discarding blocks of data.

0 <= \_\_r <= \_\_p

Definition at line 839 of file random.h.

#### 5.700.2 Member Typedef Documentation

##### 5.700.2.1 result\_type

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
typedef _RandomNumberEngine::result_type std::discard\_block\_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

Definition at line 842 of file random.h.

### 5.700.3 Constructor & Destructor Documentation

#### 5.700.3.1 `discard_block_engine()` [1/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine ( ) [inline]
```

Constructs a default `discard_block_engine` engine.

The underlying engine is default constructed as well.

Definition at line 857 of file `random.h`.

#### 5.700.3.2 `discard_block_engine()` [2/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
    const _RandomNumberEngine & __rng ) [inline], [explicit]
```

Copy constructs a `discard_block_engine` engine.

Copies an existing base class random number generator.

##### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 867 of file `random.h`.

#### 5.700.3.3 `discard_block_engine()` [3/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
    _RandomNumberEngine && __rng ) [inline], [explicit]
```

Move constructs a `discard_block_engine` engine.

Copies an existing base class random number generator.

##### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 877 of file random.h.


#### 5.700.3.4 discard\_block\_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine (
    result_type __s ) [inline], [explicit]
```

Seed constructs a discard\_block\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

##### Parameters

 __s	A seed value for the base class engine.
--	---


Definition at line 887 of file random.h.

#### 5.700.3.5 discard\_block\_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_
_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>
std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine (
    _Sseq & __q ) [inline], [explicit]
```

Generator construct a discard\_block\_engine engine.

##### Parameters

 __q	A seed sequence.
--	------------------

Definition at line 900 of file random.h.

### 5.700.4 Member Function Documentation

#### 5.700.4.1 base()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
const _RandomNumberEngine& std::discard_block_engine<_RandomNumberEngine, __p, __r>::base ( )
const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 944 of file random.h.

#### 5.700.4.2 `discard()`

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r >::discard (
    unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Definition at line 965 of file random.h.

#### 5.700.4.3 `max()`

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r >::max ( )
[inline], [static]
```

Gets the maximum value in the generated random number range.

Definition at line 958 of file random.h.

References `std::max()`.

#### 5.700.4.4 `min()`

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r >::min ( )
[inline], [static]
```

Gets the minimum value in the generated random number range.

Definition at line 951 of file random.h.

References `std::min()`.

#### 5.700.4.5 operator()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
discard_block_engine<_RandomNumberEngine, __p, __r>::result_type std::discard_block_engine<_↵
RandomNumberEngine, __p, __r>::operator() ( )
```

Gets the next value in the generated random number sequence.

Definition at line 685 of file bits/random.tcc.

#### 5.700.4.6 seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed ( ) [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 909 of file random.h.

#### 5.700.4.7 seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed (
    result_type __s ) [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 920 of file random.h.

#### 5.700.4.8 seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq >
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed (
    _Sseq & __q ) [inline]
```

Reseeds the discard\_block\_engine object with the given seed sequence.

##### Parameters

<code>_↵</code>	A seed generator function.
<code>_q</code>	

Definition at line 933 of file random.h.

## 5.700.5 Friends And Related Function Documentation

### 5.700.5.1 operator<<

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> & __x ) [friend]
```

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

__os	An output stream.
__x	A discard_block_engine random number generator engine.

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

### 5.700.5.2 operator==

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool operator== (
    const discard_block_engine<_RandomNumberEngine, __p, __r> & __lhs,
    const discard_block_engine<_RandomNumberEngine, __p, __r> & __rhs ) [friend]
```

Compares two discard\_block\_engine random number generator objects of the same type for equality.

#### Parameters

__lhs	A discard_block_engine random number generator object.
__rhs	Another discard_block_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 989 of file random.h.

5.700.5.3 `operator>>`

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> & __x ) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A discard_block_engine random number generator engine.

## Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.701 `std::discrete_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)



## Public Member Functions

- `template<typename _InputIterator >`  
**discrete\_distribution** (`_InputIterator __wbegin, _InputIterator __wend`)
- **discrete\_distribution** (`initializer_list< double > __wl`)
- `template<typename _Func >`  
**discrete\_distribution** (`size_t __nw, double __xmin, double __xmax, _Func __fw`)
- **discrete\_distribution** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `param_type param` () const
- `void param` (`const param_type &__param`)
- `std::vector< double > probabilities` () const
- `void reset` ()

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<` (`std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x`)
- `bool operator==` (`const discrete_distribution &__d1, const discrete_distribution &__d2`)
- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x`)

## 5.701.1 Detailed Description

```
template<typename _IntType = int>
class std::discrete_distribution< _IntType >
```

A `discrete_distribution` random number distribution.

The formula for the discrete probability mass function is

Definition at line 5171 of file `random.h`.

### 5.701.2 Member Typedef Documentation

#### 5.701.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::discrete_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 5174 of file random.h.

### 5.701.3 Member Function Documentation

#### 5.701.3.1 max()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5296 of file random.h.

References std::vector<\_Tp, \_Alloc>::empty(), and std::vector<\_Tp, \_Alloc>::size().

#### 5.701.3.2 min()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5289 of file random.h.

#### 5.701.3.3 operator>()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::discrete_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 5307 of file random.h.

#### 5.701.3.4 param() [1/2]

```
template<typename _IntType = int>
param_type std::discrete_distribution< _IntType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5274 of file random.h.

#### 5.701.3.5 param() [2/2]

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5282 of file random.h.

#### 5.701.3.6 probabilities()

```
template<typename _IntType = int>
std::vector<double> std::discrete_distribution< _IntType >::probabilities ( ) const [inline]
```

Returns the probabilities of the distribution.

Definition at line 5264 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

#### 5.701.3.7 reset()

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 5257 of file random.h.

## 5.701.4 Friends And Related Function Documentation

## 5.701.4.1 operator&lt;&lt;

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::discrete_distribution<_IntType1> & __x ) [friend]
```

Inserts a discrete\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

__os	An output stream.
__x	A discrete_distribution random number distribution.

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## 5.701.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
    const discrete_distribution<_IntType> & __d1,
    const discrete_distribution<_IntType> & __d2 ) [friend]
```

Return true if two discrete distributions have the same parameters.

Definition at line 5342 of file random.h.

## 5.701.4.3 operator&gt;&gt;

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::discrete_distribution<_IntType1> & __x ) [friend]
```

Extracts a discrete\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

<a href="#"><code>__is</code></a>	An input stream.
<a href="#"><code>__x</code></a>	A discrete_distribution random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.702 `std::discrete_distribution<_IntType>::param_type` Struct Reference

## Public Types

- typedef [discrete\\_distribution](#)<\_IntType> **distribution\_type**

## Public Member Functions

- template<typename \_InputIterator >  
**param\_type** (\_InputIterator \_\_wbegin, \_InputIterator \_\_wend)
- **param\_type** ([initializer\\_list](#)< double > \_\_wil)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default
- [std::vector](#)< double > **probabilities** () const

## Friends

- class **discrete\_distribution**<\_IntType>
- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.702.1 Detailed Description

```
template<typename _IntType = int>
struct std::discrete_distribution<_IntType>::param_type
```

Parameter type.

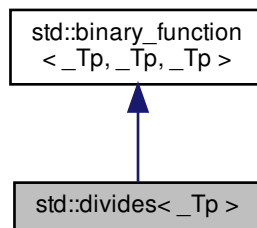
Definition at line 5181 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.703 std::divides< \_Tp > Struct Template Reference

Inheritance diagram for std::divides< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `_GLIBCXX14_CONSTEXPR _Tp operator()(const _Tp &__x, const _Tp &__y) const`

#### 5.703.1 Detailed Description

```
template<typename _Tp>  
struct std::divides< _Tp >
```

One of the [math functors](#).

Definition at line 156 of file `stl_function.h`.

#### 5.703.2 Member Typedef Documentation

#### 5.703.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.703.2.2 result\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.703.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.704 std::divides< void > Struct Template Reference

#### Public Types

- typedef \_\_is\_transparent **is\_transparent**

#### Public Member Functions

- template<typename \_Tp, typename \_Up >  
\_GLIBCXX14\_CONSTEXPR auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t)/std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t)/std::forward< \_Up >(\_\_u))

## 5.704.1 Detailed Description

```
template<>
struct std::divides< void >
```

One of the [math functors](#).

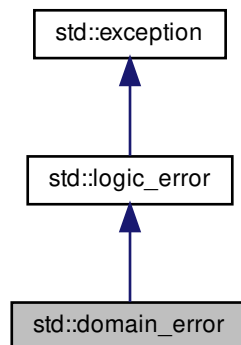
Definition at line 275 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.705 std::domain\_error Class Reference

Inheritance diagram for std::domain\_error:



## Public Member Functions

- **domain\_error** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **domain\_error** (const char \*) \_GLIBCXX\_TXN\_SAFE
- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.705.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 147 of file stdexcept.



## 5.705.2 Member Function Documentation

### 5.705.2.1 `what()`

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.706 `std::enable_if< bool, _Tp >` Struct Template Reference

### 5.706.1 Detailed Description

```
template<bool, typename _Tp = void>  
struct std::enable_if< bool, _Tp >
```

Define a member typedef `type` only if a boolean constant is true.

Definition at line 1906 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.707 `std::enable_shared_from_this< _Tp >` Class Template Reference

### Public Member Functions

- [shared\\_ptr< \\_Tp >](#) **shared\_from\_this** ()
- [shared\\_ptr< const \\_Tp >](#) **shared\_from\_this** () const
- [weak\\_ptr< \\_Tp >](#) **weak\_from\_this** () noexcept
- [weak\\_ptr< const \\_Tp >](#) **weak\_from\_this** () const noexcept

### Protected Member Functions

- **enable\_shared\_from\_this** (const [enable\\_shared\\_from\\_this](#) &) noexcept
- [enable\\_shared\\_from\\_this](#) & **operator=** (const [enable\\_shared\\_from\\_this](#) &) noexcept

## Friends

- const [enable\\_shared\\_from\\_this](#) \* [\\_\\_enable\\_shared\\_from\\_this\\_base](#) (const [\\_\\_shared\\_count](#)<> &, const [enable\\_shared\\_from\\_this](#) \* \_\_p)
- template<typename , [\\_Lock\\_policy](#) >  
class [\\_\\_shared\\_ptr](#)

## 5.707.1 Detailed Description

```
template<typename _Tp>
class std::enable_shared_from_this< _Tp >
```

Base class allowing use of member function `shared_from_this`.

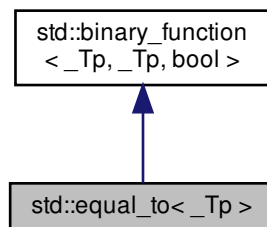
Definition at line 639 of file `bits/shared_ptr.h`.

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.708 std::equal\_to&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for `std::equal_to< _Tp >`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_GLIBCXX14_CONSTEXPR bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.708.1 Detailed Description

```
template<typename _Tp>
struct std::equal_to<_Tp>
```

One of the [comparison functors](#).

Definition at line 331 of file `stl_function.h`.

### 5.708.2 Member Typedef Documentation

#### 5.708.2.1 `first_argument_type`

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.708.2.2 `result_type`

```
typedef bool std::binary\_function< _Tp , _Tp , bool >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 5.708.2.3 `second_argument_type`

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.709 std::equal\_to&lt; void &gt; Struct Template Reference

## Public Types

- typedef \_\_is\_transparent **is\_transparent**

## Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) == std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) == std::forward< \_Up >(\_\_u))

## 5.709.1 Detailed Description

```
template<>
struct std::equal_to< void >
```

One of the [comparison functors](#).

Definition at line 464 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.710 std::error\_code Struct Reference

## Public Member Functions

- **error\_code** (int \_\_v, const error\_category &\_\_cat) noexcept
- template<typename \_ErrorCodeEnum, typename = typename enable\_if<is\_error\_code\_enum<\_ErrorCodeEnum>::value::type>  
**error\_code** (\_ErrorCodeEnum \_\_e) noexcept
- void **assign** (int \_\_v, const error\_category &\_\_cat) noexcept
- const error\_category & **category** () const noexcept
- void **clear** () noexcept
- [error\\_condition](#) **default\_error\_condition** () const noexcept
- \_GLIBCXX\_DEFAULT\_ABI\_TAG [string message](#) () const
- **operator bool** () const noexcept
- template<typename \_ErrorCodeEnum >  
[enable\\_if< is\\_error\\_code\\_enum < \\_ErrorCodeEnum >::value, error\\_code & >::type](#) **operator=** (\_ErrorCodeEnum \_\_e) noexcept
- int **value** () const noexcept

## Friends

- class **hash< error\_code >**

### 5.710.1 Detailed Description

error\_code

Definition at line 146 of file system\_error.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.711 std::error\_condition Struct Reference

### Public Member Functions

- **error\_condition** (int \_\_v, const error\_category &\_\_cat) noexcept
- template<typename \_ErrorConditionEnum , typename = typename enable\_if<is\_error\_condition\_enum<\_ErrorConditionEnum>&↳::value>::type>  
**error\_condition** (\_ErrorConditionEnum \_\_e) noexcept
- void **assign** (int \_\_v, const error\_category &\_\_cat) noexcept
- const error\_category & **category** () const noexcept
- void **clear** () noexcept
- \_GLIBCXX\_DEFAULT\_ABI\_TAG [string message](#) () const
- **operator bool** () const noexcept
- template<typename \_ErrorConditionEnum >  
[enable\\_if< is\\_error\\_condition\\_enum< \\_ErrorConditionEnum >::value, \[error\\\_condition\]\(#\) & >::type](#) **operator=** (\_↳ErrorConditionEnum \_\_e) noexcept
- int **value** () const noexcept

### 5.711.1 Detailed Description

error\_condition

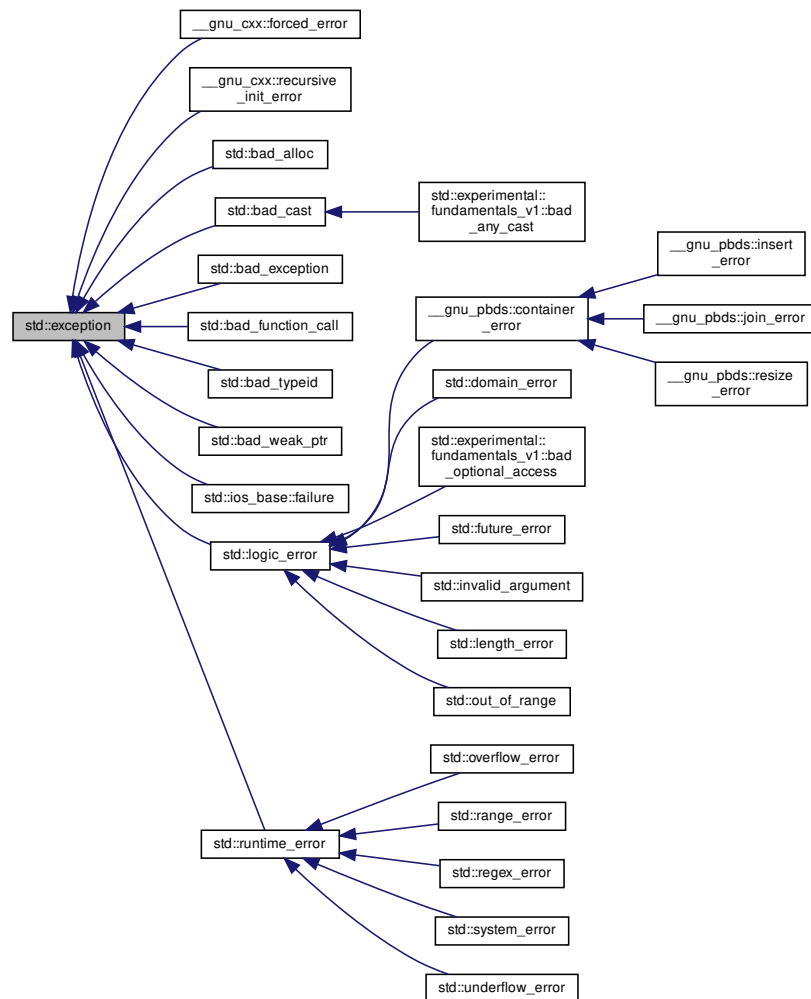
Definition at line 224 of file system\_error.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.712 std::exception Class Reference

Inheritance diagram for std::exception:



## Public Member Functions

- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.712.1 Detailed Description

Base class for all library exceptions.

This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 60 of file exception.h.

The documentation for this class was generated from the following file:

- [exception.h](#)

## 5.713 std::experimental::filesystem::v1::path Class Reference

Inherited by std::experimental::filesystem::v1::path::\_Cmpt.

### Classes

- class [iterator](#)

### Public Types

- typedef [iterator](#) **const\_iterator**
- typedef [std::basic\\_string](#)< value\_type > **string\_type**
- typedef char **value\_type**

### Public Member Functions

- **path** (const [path](#) &\_\_p)=default
- **path** ([path](#) &&\_\_p) noexcept
- **path** ([string\\_type](#) &&\_\_source)
- template<typename \_Source , typename \_Require = \_Path<\_Source>>  
**path** (\_Source const &\_\_source)
- template<typename \_InputIterator , typename \_Require = \_Path<\_InputIterator, \_InputIterator>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source , typename \_Require = \_Path<\_Source> , typename \_Require2 = \_\_value\_type\_is\_char<\_Source>>  
**path** (\_Source const &\_\_source, const [locale](#) &\_\_loc)
- template<typename \_InputIterator , typename \_Require = \_Path<\_InputIterator, \_InputIterator> , typename \_Require2 = \_\_value\_type\_is\_char<\_InputIterator>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, const [locale](#) &\_\_loc)
- template<typename \_Source >  
\_Path<\_Source> & **append** (\_Source const &\_\_source)
- template<typename \_InputIterator >  
\_Path<\_InputIterator, \_InputIterator> & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source >  
\_Path<\_Source> & **assign** (\_Source const &\_\_source)
- template<typename \_InputIterator >  
\_Path<\_InputIterator, \_InputIterator> & **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- const value\_type \* **c\_str** () const noexcept
- void **clear** () noexcept
- int **compare** (const [path](#) &\_\_p) const noexcept
- template<typename \_Source >  
\_Path<\_Source> & **concat** (\_Source const &\_\_x)

- template<typename \_InputIterator >  
\_Path< \_InputIterator, \_InputIterator > & **concat** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - bool **empty** () const noexcept
  - bool **has\_filename** () const
  - bool **has\_parent\_path** () const
  - bool **has\_relative\_path** () const
  - bool **has\_root\_directory** () const
  - bool **has\_root\_name** () const
  - bool **has\_root\_path** () const
  - bool **is\_absolute** () const
  - bool **is\_relative** () const
  - const [string\\_type](#) & **native** () const noexcept
  - **operator string\_type** () const
  - template<typename \_Source >  
\_Path< \_Source > & **operator+=** (\_Source const &\_\_x)
  - [path](#) & **operator/=** (const [path](#) &\_\_p)
  - template<class \_Source >  
\_Path< \_Source > & **operator/=** (\_Source const &\_\_source)
  - [path](#) & **operator=** (const [path](#) &\_\_p)=default
  - template<typename \_Source >  
\_Path< \_Source > & **operator=** (\_Source const &\_\_source)
  - [path](#) **parent\_path** () const
  - [path](#) **relative\_path** () const
  - [path](#) & **remove\_filename** ()
  - [path](#) & **replace\_extension** (const [path](#) &\_\_replacement=[path](#)())
  - [path](#) & **replace\_filename** (const [path](#) &\_\_replacement)
  - [path](#) **root\_directory** () const
  - [path](#) **root\_name** () const
  - [path](#) **root\_path** () const
- 
- [path](#) & **operator=** ([path](#) &&\_\_p) noexcept
  - [path](#) & **operator=** ([string\\_type](#) &&\_\_source)
  - [path](#) & **assign** ([string\\_type](#) &&\_\_source)
  - [path](#) & **operator+=** (const [path](#) &\_\_x)
  - [path](#) & **operator+=** (const [string\\_type](#) &\_\_x)
  - [path](#) & **operator+=** (const value\_type \*\_\_x)
  - [path](#) & **operator+=** (value\_type \_\_x)
  - [path](#) & **operator+=** (basic\_string\_view< value\_type > \_\_x)
  - template<typename \_CharT >  
\_Path< \_CharT \*, \_CharT \* > & **operator+=** (\_CharT \_\_x)
  - [path](#) & **make\_preferred** ()
  - void **swap** ([path](#) &\_\_rhs) noexcept
  - template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>>  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Allocator > **string** (const \_Allocator &\_\_a=\_Allocator()) const
  - [std::string](#) **string** () const
  - [std::wstring](#) **wstring** () const
  - [std::string](#) **u8string** () const
  - [std::u16string](#) **u16string** () const



- [std::u32string](#) **u32string** () const
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>>`  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Allocator > **generic\_string** (const \_Allocator &\_\_a=\_Allocator()) const
- [std::string](#) **generic\_string** () const
- [std::wstring](#) **generic\_wstring** () const
- [std::string](#) **generic\_u8string** () const
- [std::u16string](#) **generic\_u16string** () const
- [std::u32string](#) **generic\_u32string** () const
- `int` **compare** (const [string\\_type](#) &\_\_s) const
- `int` **compare** (const value\_type \*\_\_s) const
- `int` **compare** (const basic\_string\_view< value\_type > \_\_s) const
- [path](#) **filename** () const
- [path](#) **stem** () const
- [path](#) **extension** () const
- `bool` **has\_stem** () const
- `bool` **has\_extension** () const
- [iterator](#) **begin** () const
- [iterator](#) **end** () const

#### Static Public Attributes

- `static constexpr value_type` **preferred\_separator**

#### 5.713.1 Detailed Description

A filesystem path.

Definition at line 80 of file `experimental/bits/fs_path.h`.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

#### 5.714 std::experimental::filesystem::v1::path::iterator Class Reference

##### Public Types

- using **difference\_type** = `std::ptrdiff_t`
- using **iterator\_category** = [std::bidirectional\\_iterator\\_tag](#)
- using **pointer** = `const` [path](#) \*
- using **reference** = `const` [path](#) &
- using **value\_type** = [path](#)

## Public Member Functions

- `iterator` (const `iterator` &)=default
- `iterator operator++` (int)
- `iterator operator--` (int)
- `pointer operator->` () const
- `iterator & operator=` (const `iterator` &)=default

## Friends

- bool `operator!=` (const `iterator` &\_\_lhs, const `iterator` &\_\_rhs)
- bool `operator==` (const `iterator` &\_\_lhs, const `iterator` &\_\_rhs)
- class `path`
- `reference operator*` () const
- `iterator & operator++` ()
- `iterator & operator--` ()

## 5.714.1 Detailed Description

An iterator for the components of a path.

Definition at line 718 of file `experimental/bits/fs_path.h`.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

5.715 `std::experimental::fundamentals_v1::_Has_addressof<_Tp>` Struct Template Reference

Inherits type< `_Has_addressof_mem<_Tp>`, `_Has_addressof_free<_Tp>` >.

## 5.715.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v1::_Has_addressof<_Tp>
```

Trait that detects the presence of an overloaded unary operator&.

Practically speaking this detects the presence of such an operator when called on a const-qualified lvalue (e.g. `declval<const _Tp&>().operator&()`).

Definition at line 162 of file `optional`.

The documentation for this struct was generated from the following file:

- [optional](#)

## 5.716 `std::experimental::fundamentals_v1::_Optional_base< _Tp, _ShouldProvideDestructor >` Class Template Reference

### Public Member Functions

- `constexpr _Optional_base (nullopt_t) noexcept`
- `template<typename... _Args>`  
`constexpr _Optional_base (in_place_t, _Args &&... __args)`
- `template<typename _Up, typename... _Args, enable_if_t< is_constructible< _Tp, initializer_list< _Up > &, _Args &&... >::value, int > ...>`  
`constexpr _Optional_base (in_place_t, initializer_list< _Up > __il, _Args &&... __args)`
- `_Optional_base (const _Optional_base &__other)`
- `_Optional_base (_Optional_base &&__other) noexcept(is_nothrow_move_constructible< _Tp >())`
- `_Optional_base & operator= (const _Optional_base &__other)`
- `_Optional_base & operator= (_Optional_base &&__other) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp >>())`

### Protected Member Functions

- `template<typename... _Args>`  
`void _M_construct (_Args &&... __args) noexcept(is_nothrow_constructible< _Stored_type, _Args... >())`
- `void _M_destruct ()`
- `constexpr _Tp & _M_get () noexcept`
- `constexpr const _Tp & _M_get () const noexcept`
- `constexpr bool _M_is_engaged () const noexcept`
- `void _M_reset ()`

#### 5.716.1 Detailed Description

```
template<typename _Tp, bool _ShouldProvideDestructor = !is_trivially_destructible< _Tp >::value>
class std::experimental::fundamentals_v1::_Optional_base< _Tp, _ShouldProvideDestructor >
```

Class template that holds the necessary state for [Optional values](#) and that has the responsibility for construction and the special members.

Such a separate base class template is necessary in order to conditionally enable the special members (e.g. copy/move constructors). Note that this means that `_Optional_base` implements the functionality for copy and move assignment, but not for converting assignment.

#### See also

`optional`, `_Enable_special_members`

Definition at line 202 of file `optional`.

The documentation for this class was generated from the following file:

- [optional](#)

## 5.717 std::experimental::fundamentals\_v1::\_Optional\_base&lt;\_Tp, false &gt; Class Template Reference

## Public Member Functions

- constexpr **\_Optional\_base** ([nullopt\\_t](#)) noexcept
- template<typename... \_Args>  
constexpr **\_Optional\_base** ([in\\_place\\_t](#), \_Args &&... \_\_args)
- template<typename \_Up, typename... \_Args, enable\_if\_t< is\_constructible<\_Tp, initializer\_list<\_Up> &, \_Args &&... >::value, int > ...>  
constexpr **\_Optional\_base** ([in\\_place\\_t](#), [initializer\\_list](#)<\_Up> \_\_il, \_Args &&... \_\_args)
- **\_Optional\_base** (const [\\_Optional\\_base](#) &\_\_other)
- **\_Optional\_base** ([\\_Optional\\_base](#) &&\_\_other) noexcept([is\\_nothrow\\_move\\_constructible](#)<\_Tp>())
- [\\_Optional\\_base](#) & **operator=** (const [\\_Optional\\_base](#) &\_\_other)
- [\\_Optional\\_base](#) & **operator=** ([\\_Optional\\_base](#) &&\_\_other) noexcept(\_\_and\_< [is\\_nothrow\\_move\\_constructible](#)<\_Tp>, [is\\_nothrow\\_move\\_assignable](#)<\_Tp>>())

## Protected Member Functions

- template<typename... \_Args>  
void **\_M\_construct** (\_Args &&... \_\_args) noexcept([is\\_nothrow\\_constructible](#)<\_Stored\_type, \_Args... >())
- void **\_M\_destruct** ()
- \_Tp & **\_M\_get** () noexcept
- constexpr const \_Tp & **\_M\_get** () const noexcept
- constexpr bool **\_M\_is\_engaged** () const noexcept
- void **\_M\_reset** ()

## 5.717.1 Detailed Description

```
template<typename _Tp>
class std::experimental::fundamentals_v1::_Optional_base<_Tp, false >
```

Partial specialization that is exactly identical to the primary template save for not providing a destructor, to fulfill triviality requirements.

Definition at line 345 of file optional.

The documentation for this class was generated from the following file:

- [optional](#)

## 5.718 std::experimental::fundamentals\_v1::any Class Reference

### Public Member Functions

- [any](#) () noexcept
- [any](#) (const [any](#) &\_\_other)
- [any](#) ([any](#) &&\_\_other) noexcept
- template<typename \_ValueType , typename \_Tp = \_Decay<\_ValueType>, typename \_Mgr = \_Manager<\_Tp>, typename enable\_if< is\_constructible< \_Tp, \_ValueType && >::value, bool >::type = true>  
[any](#) (\_ValueType &&\_\_value)
- template<typename \_ValueType , typename \_Tp = \_Decay<\_ValueType>, typename \_Mgr = \_Manager<\_Tp>, typename enable\_if< !is\_constructible< \_Tp, \_ValueType && >::value, bool >::type = false>  
[any](#) (\_ValueType &&\_\_value)
- [~any](#) ()
- void [clear](#) () noexcept
- bool [empty](#) () const noexcept
- [any](#) & [operator=](#) (const [any](#) &\_\_rhs)
- [any](#) & [operator=](#) ([any](#) &&\_\_rhs) noexcept
- template<typename \_ValueType >  
[enable\\_if\\_t](#)<!is\_same< [any](#), decay\_t< \_ValueType > >::value, [any](#) & > [operator=](#) (\_ValueType &&\_\_rhs)
- void [swap](#) ([any](#) &\_\_rhs) noexcept
- const [type\\_info](#) & [type](#) () const noexcept

### Static Public Member Functions

- template<typename \_Tp >  
static constexpr bool [\\_\\_is\\_valid\\_cast](#) ()

### Friends

- template<typename \_Tp >  
[enable\\_if\\_t](#)< [is\\_object](#)< \_Tp >::value, void \* > [\\_\\_any\\_caster](#) (const [any](#) \*\_\_any)

#### 5.718.1 Detailed Description

A type-safe container of any type.

An [any](#) object's state is either empty or it stores a contained object of CopyConstructible type.

Definition at line 87 of file [any](#).

#### 5.718.2 Constructor & Destructor Documentation

**5.718.2.1 any()** [1/5]

```
std::experimental::fundamentals_v1::any::any ( ) [inline], [noexcept]
```

Default constructor, creates an empty object.

Definition at line 126 of file any.

**5.718.2.2 any()** [2/5]

```
std::experimental::fundamentals_v1::any::any (
    const any & __other ) [inline]
```

Copy constructor, copies the state of \_\_other.

Definition at line 129 of file any.

**5.718.2.3 any()** [3/5]

```
std::experimental::fundamentals_v1::any::any (
    any && __other ) [inline], [noexcept]
```

Move constructor, transfer the state from \_\_other.

**Postcondition**

\_\_other.empty() (this postcondition is a GNU extension)

Definition at line 146 of file any.

**5.718.2.4 any()** [4/5]

```
template<typename _ValueType , typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
std::experimental::fundamentals_v1::any::any (
    _ValueType && __value ) [inline]
```

Construct with a copy of \_\_value as the contained object.

Definition at line 163 of file any.

**5.718.2.5 any()** [5/5]

```
template<typename _ValueType , typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,  
typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>  
std::experimental::fundamentals_v1::any::any (   
    _ValueType && __value ) [inline]
```

Construct with a copy of `__value` as the contained object.

Definition at line 176 of file `any`.

**5.718.2.6 ~any()**

```
std::experimental::fundamentals_v1::any::~~any ( ) [inline]
```

Destructor, calls `clear()`

Definition at line 185 of file `any`.

**5.718.3 Member Function Documentation****5.718.3.1 clear()**

```
void std::experimental::fundamentals_v1::any::clear ( ) [inline], [noexcept]
```

If not empty, destroy the contained object.

Definition at line 227 of file `any`.

**5.718.3.2 empty()**

```
bool std::experimental::fundamentals_v1::any::empty ( ) const [inline], [noexcept]
```

Reports whether there is a contained object or not.

Definition at line 269 of file `any`.

**5.718.3.3 operator=()** [1/3]

```
any& std::experimental::fundamentals_v1::any::operator= (
    const any & __rhs ) [inline]
```

Copy the state of another object.

Definition at line 190 of file any.

**5.718.3.4 operator=()** [2/3]

```
any& std::experimental::fundamentals_v1::any::operator= (
    any && __rhs ) [inline], [noexcept]
```

Move assignment operator.

**Postcondition**

`__rhs.empty()` (not guaranteed for other implementations)

Definition at line 201 of file any.

**5.718.3.5 operator=()** [3/3]

```
template<typename _ValueType >
enable_if_t<!is_same<any, decay_t<_ValueType> >::value, any&> std::experimental::fundamentals_v1::any::operator= (
    _ValueType && __rhs ) [inline]
```

Store a copy of `__rhs` as the contained object.

Definition at line 218 of file any.

**5.718.3.6 swap()**

```
void std::experimental::fundamentals_v1::any::swap (
    any & __rhs ) [inline], [noexcept]
```

Exchange state with another object.

Definition at line 237 of file any.



### 5.718.3.7 `type()`

```
const type\_info& std::experimental::fundamentals_v1::any::type ( ) const [inline], [noexcept]
```

The `typeid` of the contained object, or `typeid(void)` if empty.

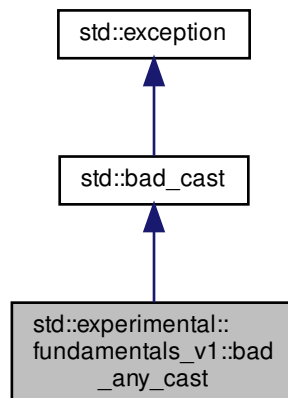
Definition at line 273 of file `any`.

The documentation for this class was generated from the following file:

- [any](#)

## 5.719 `std::experimental::fundamentals_v1::bad_any_cast` Class Reference

Inheritance diagram for `std::experimental::fundamentals_v1::bad_any_cast`:



### Public Member Functions

- virtual const char \* [what](#) ( ) const noexcept

### 5.719.1 Detailed Description

Exception class thrown by a failed `any_cast`.

Definition at line 66 of file `any`.

## 5.719.2 Member Function Documentation

5.719.2.1 `what()`

```
virtual const char* std::experimental::fundamentals_v1::bad_any_cast::what ( ) const [inline],
[virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::bad\\_cast](#).

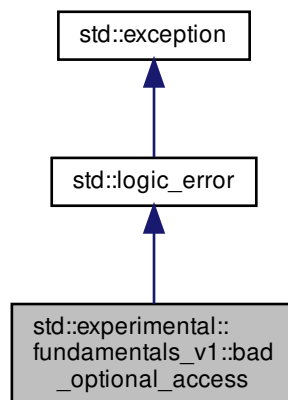
Definition at line 69 of file `any`.

The documentation for this class was generated from the following file:

- [any](#)

5.720 `std::experimental::fundamentals_v1::bad_optional_access` Class Reference

Inheritance diagram for `std::experimental::fundamentals_v1::bad_optional_access`:



## Public Member Functions

- **`bad_optional_access`** (`const char * __arg`)
- virtual `const char * what ( ) const` `_GLIBCXX_TXN_SAFE_DYN` `noexcept`

### 5.720.1 Detailed Description

Exception class thrown when a disengaged optional object is dereferenced.

Definition at line 115 of file optional.

### 5.720.2 Member Function Documentation

#### 5.720.2.1 what()

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [optional](#)

## 5.721 std::experimental::fundamentals\_v1::basic\_string\_view<\_CharT,\_Traits> Class Template Reference

### Public Types

- using **const\_iterator** = const \_CharT \*
- using **const\_pointer** = const \_CharT \*
- using **const\_reference** = const \_CharT &
- using **const\_reverse\_iterator** = [std::reverse\\_iterator](#)< const\_iterator >
- using **difference\_type** = ptrdiff\_t
- using **iterator** = const\_iterator
- using **pointer** = const \_CharT \*
- using **reference** = const \_CharT &
- using **reverse\_iterator** = [const\\_reverse\\_iterator](#)
- using **size\_type** = size\_t
- using **traits\_type** = \_Traits
- using **value\_type** = \_CharT

## Public Member Functions

- constexpr **basic\_string\_view** (const [basic\\_string\\_view](#) &) noexcept=default
- template<typename \_Allocator >  
**basic\_string\_view** (const [basic\\_string](#)<\_CharT, \_Traits, \_Allocator> &\_\_str) noexcept
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str)
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str, size\_type \_\_len)
- constexpr const \_CharT & **at** (size\_type \_\_pos) const
- constexpr const \_CharT & **back** () const
- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr int **compare** ([basic\\_string\\_view](#) \_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (const \_CharT \*\_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str, size\_type \_\_n2) const
- size\_type **copy** (\_CharT \*\_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- constexpr const \_CharT \* **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr const \_CharT & **front** () const
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- template<typename \_Allocator >  
**operator basic\_string**<\_CharT, \_Traits, \_Allocator> () const
- [basic\\_string\\_view](#) & **operator=** (const [basic\\_string\\_view](#) &) noexcept=default

- constexpr const \_CharT & **operator[]** (size\_type \_\_pos) const
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n)
- constexpr void **remove\_suffix** (size\_type \_\_n)
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr size\_type **rfind** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr [basic\\_string\\_view](#) **substr** (size\_type \_\_pos=0, size\_type \_\_n=npos) const
- constexpr void **swap** ([basic\\_string\\_view](#) &\_\_sv) noexcept
- template<typename \_Allocator = std::allocator<\_CharT>>  
[basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **to\_string** (const \_Allocator &\_\_alloc=\_Allocator()) const

#### Static Public Attributes

- static constexpr size\_type **npos**

#### 5.721.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>
```

A non-owning reference to a string.

#### Template Parameters

<code>_CharT</code>	Type of character
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> .

A `basic_string_view` looks like this:

```
_CharT*    _M_str
size_t     _M_len
```

Definition at line 74 of file `string_view`.

The documentation for this class was generated from the following files:

- [string\\_view](#)
- [experimental/bits/string\\_view.tcc](#)

## 5.722 std::experimental::fundamentals\_v1::in\_place\_t Struct Reference

#### 5.722.1 Detailed Description

Tag type for in-place construction.

Definition at line 85 of file optional.

The documentation for this struct was generated from the following file:

- [optional](#)

## 5.723 std::experimental::fundamentals\_v1::nullopt\_t Struct Reference

### Public Types

- enum **\_Construct** { **\_Token** }

### Public Member Functions

- constexpr **nullopt\_t** (\_Construct)

#### 5.723.1 Detailed Description

Tag type to disengage optional objects.

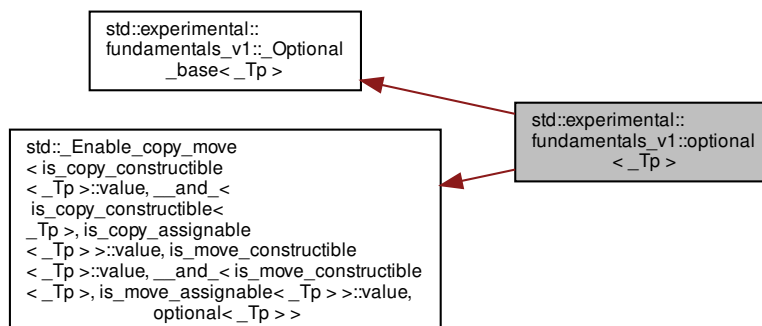
Definition at line 92 of file optional.

The documentation for this struct was generated from the following file:

- [optional](#)

## 5.724 std::experimental::fundamentals\_v1::optional<\_Tp> Class Template Reference

Inheritance diagram for std::experimental::fundamentals\_v1::optional<\_Tp>:



## Public Types

- using **value\_type** = **\_Tp**

## Public Member Functions

- template<typename **\_Up** = **\_Tp**, enable\_if\_t< \_\_and< \_\_not< is\_same< optional< **\_Tp** >, decay\_t< **\_Up** >>>, is\_constructible< **\_Tp**, **\_Up** && >, is\_convertible< **\_Up** &&, **\_Tp** >>::value, bool > = true>  
constexpr **optional** (**\_Up** &&\_\_t)
- template<typename **\_Up** = **\_Tp**, enable\_if\_t< \_\_and< \_\_not< is\_same< optional< **\_Tp** >, decay\_t< **\_Up** >>>, is\_constructible< **\_Tp**, **\_Up** && >, \_\_not< is\_convertible< **\_Up** &&, **\_Tp** >>::value, bool > = false>  
constexpr **optional** (**\_Up** &&\_\_t)
- template<typename **\_Up** , enable\_if\_t< \_\_and< \_\_not< is\_same< **\_Tp**, **\_Up** >>, is\_constructible< **\_Tp**, const **\_Up** & >, is\_convertible< const **\_Up** &, **\_Tp** >, \_\_not< \_\_converts\_from\_optional< **\_Tp**, **\_Up** >>::value, bool > = true>  
constexpr **optional** (const **optional**< **\_Up** > &\_\_t)
- template<typename **\_Up** , enable\_if\_t< \_\_and< \_\_not< is\_same< **\_Tp**, **\_Up** >>, is\_constructible< **\_Tp**, const **\_Up** & >, \_\_not< is\_convertible< const **\_Up** &, **\_Tp** >>, \_\_not< \_\_converts\_from\_optional< **\_Tp**, **\_Up** >>::value, bool > = false>  
constexpr **optional** (const **optional**< **\_Up** > &\_\_t)
- template<typename **\_Up** , enable\_if\_t< \_\_and< \_\_not< is\_same< **\_Tp**, **\_Up** >>, is\_constructible< **\_Tp**, **\_Up** && >, is\_convertible< **\_Up** &&, **\_Tp** >, \_\_not< \_\_converts\_from\_optional< **\_Tp**, **\_Up** >>::value, bool > = true>  
constexpr **optional** (**optional**< **\_Up** > &&\_\_t)
- template<typename **\_Up** , enable\_if\_t< \_\_and< \_\_not< is\_same< **\_Tp**, **\_Up** >>, is\_constructible< **\_Tp**, **\_Up** && >, \_\_not< is\_convertible< **\_Up** &&, **\_Tp** >>, \_\_not< \_\_converts\_from\_optional< **\_Tp**, **\_Up** >>::value, bool > = false>  
constexpr **optional** (**optional**< **\_Up** > &&\_\_t)
- template<typename... **\_Args**>  
**enable\_if\_t**< **is\_constructible**< **\_Tp**, **\_Args** &&... >::value > **emplace** (**\_Args** &&... \_\_args)
- template<typename **\_Up** , typename... **\_Args**>  
**enable\_if\_t**< **is\_constructible**< **\_Tp**, **initializer\_list**< **\_Up** > &, **\_Args** &&... >::value > **emplace** (**initializer\_list**< **\_Up** > \_\_il, **\_Args** &&... \_\_args)
- constexpr **operator bool** () const noexcept
- constexpr const **\_Tp** & **operator\*** () const &
- constexpr **\_Tp** & **operator\*** () &
- constexpr **\_Tp** && **operator\*** () &&
- constexpr const **\_Tp** && **operator\*** () const &&
- constexpr const **\_Tp** \* **operator->** () const
- **\_Tp** \* **operator->** ()
- **optional** & **operator=** (**nullopt\_t**) noexcept
- template<typename **\_Up** = **\_Tp**>  
**enable\_if\_t**< \_\_and< \_\_not< is\_same< **optional**< **\_Tp** >, decay\_t< **\_Up** >>>, is\_constructible< **\_Tp**, **\_Up** >, \_\_not< \_\_and< is\_scalar< **\_Tp** >, is\_same< **\_Tp**, decay\_t< **\_Up** >>>, is\_assignable< **\_Tp** &, **\_Up** >>::value, **optional** & > **operator=** (**\_Up** &&\_\_u)
- template<typename **\_Up** >  
**enable\_if\_t**< \_\_and< \_\_not< is\_same< **\_Tp**, **\_Up** >>, is\_constructible< **\_Tp**, const **\_Up** & >, is\_assignable< **\_Tp** &, **\_Up** >, \_\_not< \_\_converts\_from\_optional< **\_Tp**, **\_Up** >>, \_\_not< \_\_assigns\_from\_optional< **\_Tp**, **\_Up** >>::value, **optional** & > **operator=** (const **optional**< **\_Up** > &\_\_u)
- template<typename **\_Up** >  
**enable\_if\_t**< \_\_and< \_\_not< is\_same< **\_Tp**, **\_Up** >>, is\_constructible< **\_Tp**, **\_Up** >, is\_assignable< **\_Tp** &, **\_Up** >, \_\_not< \_\_converts\_from\_optional< **\_Tp**, **\_Up** >>, \_\_not< \_\_assigns\_from\_optional< **\_Tp**, **\_Up** >>::value, **optional** & > **operator=** (**optional**< **\_Up** > &&\_\_u)
- void **swap** (**optional** &\_\_other) noexcept(**is\_nothrow\_move\_constructible**< **\_Tp** >()) &&\_\_is\_nothrow\_swappable< **\_Tp** >::value)
- constexpr const **\_Tp** & **value** () const &

- `constexpr _Tp & value () &`
- `constexpr _Tp && value () &&`
- `constexpr const _Tp && value () const &&`
- `template<typename _Up >`  
`constexpr _Tp value_or (_Up &&__u) const &`
- `template<typename _Up >`  
`_Tp value_or (_Up &&__u) &&`

#### Private Member Functions

- `void _M_construct (_Args &&... __args) noexcept(is_nothrow_constructible<_Stored_type, _Args... >())`
- `void _M_destruct ()`
- `constexpr _Tp & _M_get () noexcept`
- `constexpr const _Tp & _M_get () const noexcept`
- `constexpr bool _M_is_engaged () const noexcept`
- `void _M_reset ()`

#### Private Attributes

- `_Empty_byte _M_empty`
- `_Stored_type _M_payload`

#### 5.724.1 Detailed Description

```
template<typename _Tp>
class std::experimental::fundamentals_v1::optional<_Tp>
```

Class template for optional values.

Definition at line 81 of file optional.

The documentation for this class was generated from the following file:

- [optional](#)

#### 5.725 `std::experimental::fundamentals_v2::ostream_joiner<_DelimT, _CharT, _Traits>` Class Template Reference

#### Public Types

- `typedef _CharT char_type`
- `typedef void difference_type`
- `typedef output\_iterator\_tag iterator_category`
- `typedef basic\_ostream<_CharT, _Traits> ostream_type`
- `typedef void pointer`
- `typedef void reference`
- `typedef _Traits traits_type`
- `typedef void value_type`



### Public Member Functions

- **ostream\_joiner** ([ostream\\_type](#) &\_\_os, const \_DelimT &\_\_delimiter) noexcept(is\_nothrow\_copy\_constructible\_v<\_DelimT >)
- **ostream\_joiner** ([ostream\\_type](#) &\_\_os, \_DelimT &&\_\_delimiter) noexcept(is\_nothrow\_move\_constructible\_v<\_DelimT >)
- [ostream\\_joiner](#) & **operator\*** () noexcept
- [ostream\\_joiner](#) & **operator++** () noexcept
- [ostream\\_joiner](#) & **operator++** (int) noexcept
- template<typename \_Tp >  
[ostream\\_joiner](#) & **operator=** (const \_Tp &\_\_value)

#### 5.725.1 Detailed Description

```
template<typename _DelimT, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >
```

Output iterator that inserts a delimiter between elements.

Definition at line 57 of file experimental/iterator.

The documentation for this class was generated from the following file:

- [experimental/iterator](#)

#### 5.726 std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > > Struct Template Reference

Inherits std::Sp\_owner\_less< \_Tp, \_Tp1 >.

### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 5.726.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >
```

Partial specialization of owner\_less for shared\_ptr.

Definition at line 503 of file experimental/bits/shared\_ptr.h.

### 5.726.2 Member Typedef Documentation

#### 5.726.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.726.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.726.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

### 5.727 std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_Sp\_owner\_less< \_Tp, \_Tp1 >.

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (`const _Tp &__lhs, const _Tp &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp &__lhs, const _Tp1 &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp1 &__lhs, const _Tp &__rhs`) `const noexcept`

#### 5.727.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 509 of file `experimental/bits/shared_ptr.h`.

#### 5.727.2 Member Typedef Documentation

##### 5.727.2.1 first\_argument\_type

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 5.727.2.2 result\_type

```
typedef bool std::binary\_function< _Tp , _Tp , bool >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

## 5.727.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

## 5.728 std::experimental::fundamentals\_v2::propagate\_const&lt; \_Tp &gt; Class Template Reference

## Public Types

- typedef [remove\\_reference\\_t](#)< decltype(\*std::declval< \_Tp >))> **element\_type**

## Public Member Functions

- **propagate\_const** (const [propagate\\_const](#) &\_\_p)=delete
- constexpr **propagate\_const** ([propagate\\_const](#) &&\_\_p)=default
- template<typename \_Up , typename enable\_if< \_\_and\_< is\_constructible< \_Tp, \_Up && >, is\_convertible< \_Up &&, \_Tp >>::value, bool >::type = true>  
constexpr **propagate\_const** ([propagate\\_const](#)< \_Up > &&\_\_pu)
- template<typename \_Up , typename enable\_if< \_\_and\_< is\_constructible< \_Tp, \_Up && >, \_\_not\_< is\_convertible< \_Up &&, \_Tp >>::value, bool >::type = false>  
constexpr **propagate\_const** ([propagate\\_const](#)< \_Up > &&\_\_pu)
- template<typename \_Up , typename enable\_if< \_\_and\_< is\_constructible< \_Tp, \_Up && >, is\_convertible< \_Up &&, \_Tp >, \_\_not\_< \_\_is\_propagate\_const< typename decay< \_Up >::type >> >::value, bool >::type = true>  
constexpr **propagate\_const** (\_Up &&\_\_u)
- template<typename \_Up , typename enable\_if< \_\_and\_< is\_constructible< \_Tp, \_Up && >, \_\_not\_< is\_convertible< \_Up &&, \_Tp >>, \_\_not\_< \_\_is\_propagate\_const< typename decay< \_Up >::type >> >::value, bool >::type = false>  
constexpr **propagate\_const** (\_Up &&\_\_u)
- constexpr const element\_type \* **get** () const
- constexpr element\_type \* **get** ()
- constexpr **operator bool** () const
- template<typename \_Up = \_Tp, typename enable\_if< \_\_or\_< is\_pointer< \_Up >, is\_convertible< \_Up, const element\_type \* >>::value, bool >::type = true>  
constexpr **operator const element\_type \*** () const
- template<typename \_Up = \_Tp, typename enable\_if< \_\_or\_< is\_pointer< \_Up >, is\_convertible< \_Up, const element\_type \* >>::value, bool >::type = true>  
constexpr **operator element\_type \*** ()
- constexpr const element\_type & **operator\*** () const
- constexpr element\_type & **operator\*** ()
- constexpr const element\_type \* **operator->** () const
- constexpr element\_type \* **operator->** ()
- [propagate\\_const](#) & **operator=** (const [propagate\\_const](#) &\_\_p)=delete
- constexpr [propagate\\_const](#) & **operator=** ([propagate\\_const](#) &&\_\_p)=default
- template<typename \_Up , typename = typename enable\_if<is\_convertible<\_Up&&, \_Tp>::value>::type>  
constexpr [propagate\\_const](#) & **operator=** ([propagate\\_const](#)< \_Up > &&\_\_pu)
- template<typename \_Up , typename = typename enable\_if<\_\_and\_<is\_convertible<\_Up&&, \_Tp>, \_\_not\_<\_\_is\_propagate\_const< typename decay<\_Up>::type>> >::value>::type>  
constexpr [propagate\\_const](#) & **operator=** (\_Up &&\_\_u)
- constexpr void **swap** ([propagate\\_const](#) &\_\_pt) noexcept(\_\_is\_nothrow\_swappable< \_Tp >::value)

## Friends

- `template<typename _Up >`  
`constexpr const _Up & get_underlying (const propagate\_const< _Up > &__pt) noexcept`
- `template<typename _Up >`  
`constexpr _Up & get_underlying (propagate\_const< _Up > &__pt) noexcept`

### 5.728.1 Detailed Description

```
template<typename _Tp>
class std::experimental::fundamentals_v2::propagate_const< _Tp >
```

Const-propagating wrapper.

Definition at line 63 of file `propagate_const`.

The documentation for this class was generated from the following file:

- [propagate\\_const](#)

### 5.729 `std::exponential_distribution< _RealType >` Class Template Reference

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_RealType` [result\\_type](#)

#### Public Member Functions

- [exponential\\_distribution](#) (const [result\\_type](#) &\_\_lambda=[result\\_type](#)(1))
- **exponential\_distribution** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate (result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `_RealType lambda () const`
- `result\_type max () const`
- `result\_type min () const`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `param\_type param () const`
- `void param (const param\_type &__param)`
- `void reset ()`

## Friends

- bool `operator==` (const `exponential_distribution` &\_\_d1, const `exponential_distribution` &\_\_d2)

## 5.729.1 Detailed Description

```
template<typename _RealType = double>
class std::exponential_distribution<_RealType>
```

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

**Table 1994 Distribution Statistics**

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	<i>zero</i>
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Definition at line 4551 of file random.h.

## 5.729.2 Member Typedef Documentation

## 5.729.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::exponential_distribution<_RealType>::result_type
```

The type of the range of the distribution.

Definition at line 4554 of file random.h.

## 5.729.3 Constructor &amp; Destructor Documentation

## 5.729.3.1 exponential\_distribution()

```
template<typename _RealType = double>
std::exponential_distribution<_RealType>::exponential_distribution (
    const result_type & __lambda = result_type(1) ) [inline], [explicit]
```

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

Definition at line 4594 of file random.h.

#### 5.729.4 Member Function Documentation

##### 5.729.4.1 `lambda()`

```
template<typename _RealType = double>
_RealType std::exponential_distribution< _RealType >::lambda ( ) const [inline]
```

Returns the inverse scale parameter of the distribution.

Definition at line 4615 of file random.h.

Referenced by `std::operator<<()`.

##### 5.729.4.2 `max()`

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4644 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

##### 5.729.4.3 `min()`

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4637 of file random.h.

##### 5.729.4.4 `operator()()`

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::exponential_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 4652 of file random.h.

#### 5.729.4.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::exponential_distribution<_RealType>::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4622 of file random.h.

Referenced by std::operator>>().

#### 5.729.4.6 param() [2/2]

```
template<typename _RealType = double>
void std::exponential_distribution<_RealType>::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4630 of file random.h.

#### 5.729.4.7 reset()

```
template<typename _RealType = double>
void std::exponential_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4609 of file random.h.

### 5.729.5 Friends And Related Function Documentation



#### 5.729.5.1 operator==

```
template<typename _RealType = double>
bool operator== (
    const exponential\_distribution< _RealType > & __d1,
    const exponential\_distribution< _RealType > & __d2 ) [friend]
```

Return true if two exponential distributions have the same parameters.

Definition at line 4692 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.730 std::exponential\_distribution< \_RealType >::param\_type Struct Reference

#### Public Types

- typedef [exponential\\_distribution](#)< \_RealType > **distribution\_type**

#### Public Member Functions

- **param\_type** (\_RealType \_\_lambda=\_RealType(1))
- \_RealType **lambda** () const

#### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.730.1 Detailed Description

```
template<typename _RealType = double>
struct std::exponential_distribution< _RealType >::param_type
```

Parameter type.

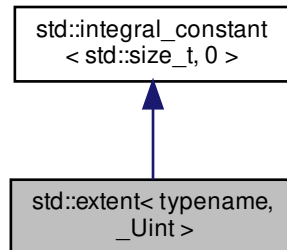
Definition at line 4561 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.731 `std::extent< typename, _UInt >` Struct Template Reference

Inheritance diagram for `std::extent< typename, _UInt >`:



## Public Types

- typedef [integral\\_constant](#)< `std::size_t`, `__v` > **type**
- typedef `std::size_t` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr `std::size_t` **value**

## 5.731.1 Detailed Description

```
template<typename, unsigned _UInt>
struct std::extent< typename, _UInt >
```

extent

Definition at line 759 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.732 std::extreme\_value\_distribution< \_RealType > Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **extreme\_value\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- bool **operator==** (const [extreme\\_value\\_distribution](#) &\_\_d1, const [extreme\\_value\\_distribution](#) &\_\_d2)

### 5.732.1 Detailed Description

```
template<typename _RealType = double>
class std::extreme_value_distribution< _RealType >
```

A extreme\_value\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 4966 of file random.h.

### 5.732.2 Member Typedef Documentation

#### 5.732.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::extreme_value_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 4969 of file random.h.

### 5.732.3 Member Function Documentation

#### 5.732.3.1 a()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::a ( ) const [inline]
```

Return the  $a$  parameter of the distribution.

Definition at line 5029 of file random.h.

Referenced by std::operator<<().

#### 5.732.3.2 b()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::b ( ) const [inline]
```

Return the  $b$  parameter of the distribution.

Definition at line 5036 of file random.h.

Referenced by std::operator<<().

#### 5.732.3.3 max()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5065 of file random.h.

References std::numeric\_limits<\_Tp>::max().

#### 5.732.3.4 min()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5058 of file random.h.

References `std::numeric_limits< _Tp >::lowest()`.

#### 5.732.3.5 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::extreme_value_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 5073 of file random.h.

#### 5.732.3.6 param() [1/2]

```
template<typename _RealType = double>
param_type std::extreme_value_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5043 of file random.h.

Referenced by `std::operator>>()`.

#### 5.732.3.7 param() [2/2]

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5051 of file `random.h`.

#### 5.732.3.8 `reset()`

```
template<typename _RealType = double>
void std::extreme_value_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 5022 of file `random.h`.

### 5.732.4 Friends And Related Function Documentation

#### 5.732.4.1 `operator==`

```
template<typename _RealType = double>
bool operator== (
    const extreme_value_distribution<_RealType> & __d1,
    const extreme_value_distribution<_RealType> & __d2 ) [friend]
```

Return true if two extreme value distributions have the same parameters.

Definition at line 5108 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.733 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference

#### Public Types

- typedef `extreme_value_distribution<_RealType>` **distribution\_type**

#### Public Member Functions

- **param\_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `_RealType a () const`
- `_RealType b () const`

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.733.1 Detailed Description

```
template<typename _RealType = double>
struct std::extreme_value_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 4976 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.734 std::fisher\_f\_distribution&lt; \_RealType &gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **fisher\_f\_distribution** (\_RealType \_\_m=\_RealType(1), \_RealType \_\_n=\_RealType(1))
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- \_RealType **n** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1 , typename \_CharT , typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::fisher\_f\_distribution< \_RealType1 > &\_\_x)
- bool operator== (const fisher\_f\_distribution &\_\_d1, const fisher\_f\_distribution &\_\_d2)
- template<typename \_RealType1 , typename \_CharT , typename \_Traits >  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::fisher\_f\_distribution< \_RealType1 > &\_\_x)

## 5.734.1 Detailed Description

```
template<typename _RealType = double>
class std::fisher_f_distribution< _RealType >
```

A fisher\_f\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 3000 of file random.h.

## 5.734.2 Member Typedef Documentation

## 5.734.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::fisher_f_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 3003 of file random.h.

## 5.734.3 Member Function Documentation



#### 5.734.3.1 max()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3099 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

#### 5.734.3.2 min()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3092 of file random.h.

#### 5.734.3.3 operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::fisher_f_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 3107 of file random.h.

#### 5.734.3.4 param() [1/2]

```
template<typename _RealType = double>
param_type std::fisher_f_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3077 of file random.h.

#### 5.734.3.5 param() [2/2]

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3085 of file random.h.

## 5.734.3.6 reset()

```
template<typename _RealType = double>
void std::fisher_f_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 3056 of file random.h.

References `std::gamma_distribution<_RealType>::reset()`.

## 5.734.4 Friends And Related Function Documentation

## 5.734.4.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::fisher_f_distribution<_RealType1> & __x ) [friend]
```

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>fisher_f_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

## 5.734.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
```

```
const fisher_f_distribution< _RealType > & __d1,
const fisher_f_distribution< _RealType > & __d2 ) [friend]
```

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3155 of file random.h.

#### 5.734.4.3 operator>>

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::fisher_f_distribution< _RealType1 > & __x ) [friend]
```

Extracts a fisher\_f\_distribution random number distribution \_\_x from the input stream \_\_is.

##### Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A fisher_f_distribution random number generator engine.

##### Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 5.735 std::fisher\_f\_distribution< \_RealType >::param\_type Struct Reference

##### Public Types

- typedef `fisher_f_distribution< _RealType >` **distribution\_type**

##### Public Member Functions

- **param\_type** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- `_RealType m` () const
- `_RealType n` () const

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.735.1 Detailed Description

```
template<typename _RealType = double>  
struct std::fisher_f_distribution< _RealType >::param_type
```

Parameter type.

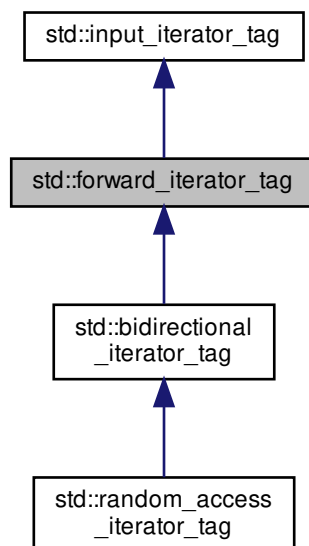
Definition at line 3010 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.736 std::forward\_iterator\_tag Struct Reference

Inheritance diagram for std::forward\_iterator\_tag:



### 5.736.1 Detailed Description

Forward iterators support a superset of input iterator operations.

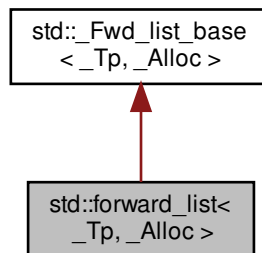
Definition at line 95 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 5.737 `std::forward_list<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::forward_list<_Tp, _Alloc>`:



#### Public Types

- `typedef _Alloc allocator_type`
- `typedef \_Fwd\_list\_const\_iterator<_Tp> const_iterator`
- `typedef \_Alloc\_traits::const\_pointer const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef \_Fwd\_list\_iterator<_Tp> iterator`
- `typedef \_Alloc\_traits::pointer pointer`
- `typedef value_type & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- [forward\\_list](#) ()=default
- [forward\\_list](#) (const \_Alloc &\_\_al) noexcept
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list, const \_Alloc &\_\_al)
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list, const \_Alloc &\_\_al) noexcept(\_Node\_alloc\_traits::\_S\_always\_equal())
- [forward\\_list](#) (size\_type \_\_n, const \_Alloc &\_\_al= \_Alloc())
- [forward\\_list](#) (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al= \_Alloc())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
[forward\\_list](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Alloc &\_\_al= \_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list)
- [forward\\_list](#) ([forward\\_list](#) &&)=default
- [forward\\_list](#) (std::initializer\_list< \_Tp > \_\_il, const \_Alloc &\_\_al= \_Alloc())
- ~[forward\\_list](#) () noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const \_Tp &\_\_val)
- void [assign](#) (std::initializer\_list< \_Tp > \_\_il)
- [iterator before\\_begin](#) () noexcept
- [const\\_iterator before\\_begin](#) () const noexcept
- [iterator begin](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [const\\_iterator cbefore\\_begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- template<typename... \_Args>  
[iterator emplace\\_after](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
void [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [iterator erase\\_after](#) (const\_iterator \_\_pos)
- [iterator erase\\_after](#) (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- [iterator insert\\_after](#) (const\_iterator \_\_pos, const \_Tp &\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, \_Tp &&\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
[iterator insert\\_after](#) (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, std::initializer\_list< \_Tp > \_\_il)
- size\_type [max\\_size](#) () const noexcept
- void [merge](#) ([forward\\_list](#) &&\_\_list)
- void [merge](#) ([forward\\_list](#) &\_\_list)
- template<typename \_Comp >  
void [merge](#) ([forward\\_list](#) &&\_\_list, \_Comp \_\_comp)
- template<typename \_Comp >  
void [merge](#) ([forward\\_list](#) &\_\_list, \_Comp \_\_comp)

- `forward_list & operator= (const forward_list &__list)`
  - `forward_list & operator= (forward_list &&__list) noexcept(_Node_alloc_traits::_S_nothrow_move())`
  - `forward_list & operator= (std::initializer_list<_Tp> __il)`
  - `void pop_front ()`
  - `void push_front (const _Tp &__val)`
  - `void push_front (_Tp &&__val)`
  - `void remove (const _Tp &__val)`
  - `template<typename _Pred>`  
`void remove_if (_Pred __pred)`
  - `void resize (size_type __sz)`
  - `void resize (size_type __sz, const value_type &__val)`
  - `void reverse () noexcept`
  - `void sort ()`
  - `template<typename _Comp>`  
`void sort (_Comp __comp)`
  - `void splice_after (const_iterator __pos, forward_list &&__list) noexcept`
  - `void splice_after (const_iterator __pos, forward_list &__list) noexcept`
  - `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __i) noexcept`
  - `void splice_after (const_iterator __pos, forward_list &__list, const_iterator __i) noexcept`
  - `void swap (forward_list &__list) noexcept`
  - `void unique ()`
  - `template<typename _BinPred>`  
`void unique (_BinPred __binary_pred)`
- 
- `void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last) noexcept`
  - `void splice_after (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last) noexcept`

#### Private Member Functions

- `template<typename... _Args>`  
`_Node * _M_create_node (_Args &&... __args)`
- `_Fwd_list_node_base * _M_erase_after (_Fwd_list_node_base * __pos)`
- `_Fwd_list_node_base * _M_erase_after (_Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `_Node_alloc_type & _M_get_Node_allocator () noexcept`
- `const _Node_alloc_type & _M_get_Node_allocator () const noexcept`
- `template<typename... _Args>`  
`_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void _M_put_node (_Node * __p)`

#### Private Attributes

- `_Fwd_list_impl _M_impl`

#### 5.737.1 Detailed Description

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
class std::forward_list<_Tp, _Alloc>
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

Definition at line 425 of file `forward_list.h`.

## 5.737.2 Constructor &amp; Destructor Documentation

5.737.2.1 `forward_list()` [1/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list ( ) [default]
```

Creates a `forward_list` with no elements.

5.737.2.2 `forward_list()` [2/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    const _Alloc & __a1 ) [inline], [explicit], [noexcept]
```

Creates a `forward_list` with no elements.

## Parameters

<code>__a1</code>	An allocator object.
-------------------	----------------------

Definition at line 468 of file `forward_list.h`.



**5.737.2.3 forward\_list()** [3/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    const forward_list< _Tp, _Alloc > & __list,
    const _Alloc & __al ) [inline]
```

Copy constructor with allocator argument.

**Parameters**

<code>__list</code>	Input list to copy.
<code>__al</code>	An allocator object.

Definition at line 477 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::begin()`, and `std::forward_list< _Tp, _Alloc >::end()`.

**5.737.2.4 forward\_list()** [4/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    forward_list< _Tp, _Alloc > && __list,
    const _Alloc & __al ) [inline], [noexcept]
```

Move constructor with allocator argument.

**Parameters**

<code>__list</code>	Input list to move.
<code>__al</code>	An allocator object.

Definition at line 505 of file forward\_list.h.

**5.737.2.5 forward\_list()** [5/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    size_type __n,
    const _Alloc & __al = _Alloc() ) [inline], [explicit]
```

Creates a forward\_list with default constructed elements.

## Parameters

<code>__n</code>	The number of elements to initially create.
<code>__al</code>	An allocator object.

This constructor creates the forward\_list with `__n` default constructed elements.

Definition at line 520 of file forward\_list.h.

## 5.737.2.6 forward\_list() [6/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    size_type __n,
    const _Tp & __value,
    const _Alloc & __al = _Alloc() ) [inline]
```

Creates a forward\_list with copies of an exemplar element.

## Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__al</code>	An allocator object.

This constructor fills the forward\_list with `__n` copies of `__value`.

Definition at line 533 of file forward\_list.h.

## 5.737.2.7 forward\_list() [7/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
std::forward_list< _Tp, _Alloc >::forward_list (
    _InputIterator __first,
    _InputIterator __last,
    const _Alloc & __al = _Alloc() ) [inline]
```

Builds a forward\_list from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__al</code>	An allocator object.

Create a `forward_list` consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is distance(`↵__first,__last`)).

Definition at line 550 of file `forward_list.h`.

#### 5.737.2.8 `forward_list()` [8/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    const forward_list< _Tp, _Alloc > & __list ) [inline]
```

The `forward_list` copy constructor.

##### Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

Definition at line 560 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::begin()`, and `std::forward_list< _Tp, _Alloc >::end()`.

#### 5.737.2.9 `forward_list()` [9/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    forward_list< _Tp, _Alloc > && ) [default]
```

The `forward_list` move constructor.

##### Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

The newly-created `forward_list` contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified `forward_list`.

#### 5.737.2.10 `forward_list()` [10/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
    std::initializer_list< _Tp > __il,
    const _Alloc & __al = _Alloc() ) [inline]
```

Builds a `forward_list` from an `initializer_list`.

## Parameters

<code>__il</code>	An initializer_list of value_type.
<code>__al</code>	An allocator object.

Create a forward\_list consisting of copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

Definition at line 584 of file forward\_list.h.

## 5.737.2.11 ~forward\_list()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc >::~~forward_list ( ) [inline], [noexcept]
```

The forward\_list dtor.

Definition at line 592 of file forward\_list.h.

## 5.737.3 Member Function Documentation

## 5.737.3.1 assign() [1/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
void std::forward_list<_Tp, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Assigns a range to a forward\_list.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a forward\_list with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the forward\_list and that the number of elements of the resulting forward\_list is the same as the number of elements assigned.

Definition at line 660 of file forward\_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::assign()`, and `std::forward_list<_Tp, _Alloc>::operator=()`.

#### 5.737.3.2 `assign()` [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::assign (
    size_type __n,
    const _Tp & __val ) [inline]
```

Assigns a given value to a `forward_list`.

##### Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a `forward_list` with `__n` copies of the given value. Note that the assignment completely changes the `forward_list`, and that the resulting `forward_list` has `__n` elements.

Definition at line 677 of file `forward_list.h`.

#### 5.737.3.3 `assign()` [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::assign (
    std::initializer_list<_Tp> __il ) [inline]
```

Assigns an `initializer_list` to a `forward_list`.

##### Parameters

<code>__il</code>	An <code>initializer_list</code> of <code>value_type</code> .
-------------------	---

Replace the contents of the `forward_list` with copies of the elements in the `initializer_list` `__il`. This is linear in `il.size()`.

Definition at line 689 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::assign()`.

#### 5.737.3.4 before\_begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 704 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::insert\_after().

#### 5.737.3.5 before\_begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 713 of file forward\_list.h.

#### 5.737.3.6 begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 721 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::forward\_list().

#### 5.737.3.7 begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 730 of file forward\_list.h.

#### 5.737.3.8 cbefore\_begin()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 766 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::emplace\_front().

#### 5.737.3.9 cbegin()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 757 of file forward\_list.h.

Referenced by std::operator<(), and std::forward\_list< \_Tp, \_Alloc >::operator=().

#### 5.737.3.10 cend()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 775 of file forward\_list.h.

Referenced by std::operator<(), and std::forward\_list< \_Tp, \_Alloc >::operator=().

#### 5.737.3.11 clear()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1088 of file forward\_list.h.

#### 5.737.3.12 emplace\_after()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
iterator std::forward_list< _Tp, _Alloc >::emplace_after (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Constructs object in forward\_list after the specified iterator.

## Parameters

<code>__pos</code>	A const_iterator into the forward_list.
<code>__args</code>	Arguments.

## Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args)...)`  after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 897 of file forward\_list.h.

5.737.3.13 `emplace_front()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
void std::forward_list< _Tp, _Alloc >::emplace_front (
    _Args &&... __args ) [inline]
```

Constructs object in forward\_list at the front of the list.

## Parameters

<code>__args</code>	Arguments.
---------------------	------------

This function will insert an object of type Tp constructed with `Tp(std::forward<Args>(args)...)`  at the front of the list. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 836 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`, and `std::forward_list< _Tp, _Alloc >::front()`.

5.737.3.14 `empty()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
bool std::forward_list< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the forward\_list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 783 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.



**5.737.3.15** `end()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 739 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`, and `std::forward_list< _Tp, _Alloc >::insert_after()`.

**5.737.3.16** `end()` [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 748 of file `forward_list.h`.

**5.737.3.17** `erase_after()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
    const_iterator __pos ) [inline]
```

Removes the element pointed to by the iterator following `pos`.

**Parameters**

<code>__pos</code>	Iterator pointing before element to be erased.
--------------------	--

**Returns**

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1000 of file `forward_list.h`.

**5.737.3.18 erase\_after()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
    const_iterator __pos,
    const_iterator __last ) [inline]
```

Remove a range of elements.

**Parameters**

<code>__pos</code>	Iterator pointing before the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

**Returns**

@ `__last`.

This function will erase the elements in the range (`__pos`,`__last`) and shorten the `forward_list` accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1023 of file `forward_list.h`.

**5.737.3.19 front()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list< _Tp, _Alloc >::front ( ) [inline]
```

Returns a read/write reference to the data at the first element of the `forward_list`.

Definition at line 800 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::emplace_front()`.

**5.737.3.20 front()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list< _Tp, _Alloc >::front ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the `forward_list`.

Definition at line 811 of file `forward_list.h`.

**5.737.3.21** `get_allocator()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 694 of file `forward_list.h`.

**5.737.3.22** `insert_after()` [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
    const_iterator __pos,
    const _Tp & __val ) [inline]
```

Inserts given value into `forward_list` after specified iterator.

**Parameters**

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__val</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 914 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

**5.737.3.23** `insert_after()` [2/4]

```
template<typename _Tp, typename _Alloc >
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
    const_iterator __pos,
    size_type __n,
    const _Tp & __val )
```

Inserts a number of copies of given data into the `forward_list`.

## Parameters

<code>__pos</code>	An iterator into the forward_list.
<code>__n</code>	Number of elements to be inserted.
<code>__val</code>	Data to be inserted.

## Returns

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 256 of file forward\_list.tcc.

## 5.737.3.24 insert\_after() [3/4]

```
template<typename _Tp , typename _Alloc >
template<typename _InputIterator , typename >
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
    const_iterator __pos,
    _InputIterator __first,
    _InputIterator __last )
```

Inserts a range into the forward\_list.

## Parameters

<code>__pos</code>	An iterator into the forward_list.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

## Returns

An iterator pointing to the last inserted element or `__pos` if `__first == __last`.

This function will insert copies of the data in the range `[__first,__last)` into the forward\_list after the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 271 of file forward\_list.tcc.

References `std::forward_list< _Tp, _Alloc >::before_begin()`, `std::forward_list< _Tp, _Alloc >::empty()`, and `std::forward_list< _Tp, _Alloc >::end()`.

**5.737.3.25** `insert_after()` [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
    const_iterator __pos,
    std::initializer_list< _Tp > __il ) [inline]
```

Inserts the contents of an `initializer_list` into `forward_list` after the specified iterator.

**Parameters**

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__il</code>	An <code>initializer_list</code> of value_type.

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the `initializer_list` `__il` into the `forward_list` before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 979 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::insert_after()`.

**5.737.3.26** `max_size()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the largest possible number of elements of `forward_list`.

Definition at line 790 of file `forward_list.h`.

References `std::allocator_traits< _Alloc >::max_size()`.

**5.737.3.27** `merge()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::merge (
    forward_list< _Tp, _Alloc > && __list ) [inline]
```

Merge sorted lists.

## Parameters

<code>__list</code>	Sorted list to merge.
---------------------	-----------------------

Assumes that both `list` and this list are sorted according to operator<(). Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Definition at line 1228 of file forward\_list.h.

## 5.737.3.28 merge() [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _Comp >
void forward_list::merge (
    forward_list< _Tp, _Alloc > && __list,
    _Comp __comp )
```

Merge sorted lists according to comparison function.

## Parameters

<code>__list</code>	Sorted list to merge.
<code>__comp</code>	Comparison function defining sort order.

Assumes that both `__list` and this list are sorted according to `comp`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to `comp()`.

Definition at line 349 of file forward\_list.tcc.

## 5.737.3.29 operator=() [1/3]

```
template<typename _Tp , typename _Alloc >
forward_list< _Tp, _Alloc > & forward_list::operator= (
    const forward_list< _Tp, _Alloc > & __list )
```

The forward\_list assignment operator.

## Parameters

<code>__list</code>	A forward_list of identical element and allocator types.
---------------------	--

All the elements of `__list` are copied.

Whether the allocator is copied depends on the allocator traits.

Definition at line 140 of file `forward_list.tcc`.

References `std::__addressof()`, `std::forward_list<_Tp, _Alloc>::cbegin()`, and `std::forward_list<_Tp, _Alloc>::cend()`.

#### 5.737.3.30 `operator=()` [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list<_Tp, _Alloc>::operator= (
    forward_list<_Tp, _Alloc> && __list ) [inline], [noexcept]
```

The `forward_list` move assignment operator.

##### Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

The contents of `__list` are moved into this `forward_list` (without copying, if the allocators permit it).

Afterwards `__list` is a valid, but unspecified `forward_list`

Whether the allocator is moved depends on the allocator traits.

Definition at line 620 of file `forward_list.h`.

#### 5.737.3.31 `operator=()` [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list<_Tp, _Alloc>::operator= (
    std::initializer_list<_Tp> __il ) [inline]
```

The `forward_list` initializer list assignment operator.

##### Parameters

<code>__il</code>	An <code>initializer_list</code> of <code>value_type</code> .
-------------------	---

Replace the contents of the `forward_list` with copies of the elements in the `initializer_list` `__il`. This is linear in `__il.size()`.

Definition at line 639 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::assign()`.

### 5.737.3.32 pop\_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::pop_front ( ) [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the forward\_list by one. Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop\_front() is called.

Definition at line 879 of file forward\_list.h.

### 5.737.3.33 push\_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::push_front (
    const _Tp & __val ) [inline]
```

Add data to the front of the forward\_list.

#### Parameters

<code>__val</code>	Data to be added.
--------------------	-------------------

This is a typical stack operation. The function creates an element at the front of the forward\_list and assigns the given data to it. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 856 of file forward\_list.h.

### 5.737.3.34 remove()

```
template<typename _Tp, typename _Alloc >
void forward_list::remove (
    const _Tp & __val )
```

Remove all elements equal to value.

#### Parameters

<code>__val</code>	The value to remove.
--------------------	----------------------

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only



erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 284 of file forward\_list.tcc.

#### 5.737.3.35 remove\_if()

```
template<typename _Tp , typename _Alloc >
template<typename _Pred >
void forward_list::remove_if (
    _Pred __pred )
```

Remove all elements satisfying a predicate.

##### Parameters

<code>__pred</code>	Unary predicate function or object.
---------------------	-------------------------------------

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 312 of file forward\_list.tcc.

#### 5.737.3.36 resize() [1/2]

```
template<typename _Tp , typename _Alloc >
void forward_list::resize (
    size_type __sz )
```

Resizes the forward\_list to the specified number of elements.

##### Parameters

<code>__sz</code>	Number of elements the forward_list should contain.
-------------------	---

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current number of elements the forward\_list is truncated, otherwise the forward\_list is extended and the new elements are default constructed.

Definition at line 182 of file forward\_list.tcc.

References `std::end()`.

**5.737.3.37** `resize()` [2/2]

```
template<typename _Tp, typename _Alloc >
void forward_list::resize (
    size_type __sz,
    const value_type & __val )
```

Resizes the forward\_list to the specified number of elements.

**Parameters**

<code>__sz</code>	Number of elements the forward_list should contain.
<code>__val</code>	Data with which new elements should be populated.

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current number of elements the forward\_list is truncated, otherwise the forward\_list is extended and new elements are populated with given data.

Definition at line 201 of file forward\_list.tcc.

**5.737.3.38** `reverse()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::reverse ( ) [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1281 of file forward\_list.h.

**5.737.3.39** `sort()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::sort ( ) [inline]
```

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1262 of file forward\_list.h.

**5.737.3.40** `sort()` [2/2]

```
template<typename _Tp , class _Alloc >
template<typename _Comp >
void forward_list::sort (
    _Comp __comp )
```

Sort the `forward_list` using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 393 of file `forward_list.tcc`.

**5.737.3.41** `splice_after()` [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
    const_iterator __pos,
    forward_list< _Tp, _Alloc > && __list ) [inline], [noexcept]
```

Insert contents of another `forward_list`.

**Parameters**

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*. *list* becomes an empty list.

Requires this  $\neq x$ .

Definition at line 1105 of file `forward_list.h`.

**5.737.3.42** `splice_after()` [2/4]

```
template<typename _Tp , typename _Alloc >
void forward_list::splice_after (
    const_iterator __pos,
    forward_list< _Tp, _Alloc > && __list,
    const_iterator __i ) [noexcept]
```

Insert element from another `forward_list`.

**Parameters**

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__i</code>	Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 239 of file forward\_list.tcc.

#### 5.737.3.43 splice\_after() [3/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
    const_iterator __pos,
    forward_list< _Tp, _Alloc > && ,
    const_iterator __before,
    const_iterator __last ) [inline], [noexcept]
```

Insert range from another forward\_list.

##### Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__before</code>	Iterator referencing before the start of range in list.
<code>__last</code>	Iterator referencing the end of range in list.

Removes elements in the range (`__before`,`__last`) and inserts them after `__pos` in constant time.

Undefined if `__pos` is in (`__before`,`__last`).

Definition at line 1149 of file forward\_list.h.

#### 5.737.3.44 splice\_after() [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
    const_iterator __pos,
    forward_list< _Tp, _Alloc > & ,
    const_iterator __before,
    const_iterator __last ) [inline], [noexcept]
```

Insert range from another forward\_list.

##### Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__before</code>	Iterator referencing before the start of range in list.
<code>__last</code>	Iterator referencing the end of range in list.

Removes elements in the range (`__before`,`__last`) and inserts them after `__pos` in constant time.

Undefined if `__pos` is in (`__before`,`__last`).

Definition at line 1154 of file `forward_list.h`.

#### 5.737.3.45 `swap()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (
    forward_list< _Tp, _Alloc > & __list ) [inline], [noexcept]
```

Swaps data with another `forward_list`.

##### Parameters

<code>__list</code>	A <code>forward_list</code> of the same element and allocator types.
---------------------	--

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1042 of file `forward_list.h`.

#### 5.737.3.46 `unique()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::unique ( ) [inline]
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1199 of file `forward_list.h`.

#### 5.737.3.47 `unique()` [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _BinPred >
void forward_list::unique (
    _BinPred __binary_pred )
```

Remove consecutive elements satisfying a predicate.

## Parameters

<code>__binary_pred</code>	Binary predicate function or object.
----------------------------	--------------------------------------

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 328 of file `forward_list.tcc`.

References `std::begin()`, and `std::end()`.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

5.738 `std::fpos<_StateT>` Class Template Reference

## Public Member Functions

- [fpos](#) ([streamoff](#) \_\_off)
- [operator streamoff](#) () const
- [fpos operator+](#) ([streamoff](#) \_\_off) const
- [fpos & operator+=](#) ([streamoff](#) \_\_off)
- [fpos operator-](#) ([streamoff](#) \_\_off) const
- [streamoff operator-](#) (const [fpos](#) &\_\_other) const
- [fpos & operator-=](#) ([streamoff](#) \_\_off)
- void [state](#) (\_StateT \_\_st)
- [\\_StateT state](#) () const

## 5.738.1 Detailed Description

```
template<typename _StateT>
class std::fpos<_StateT>
```

Class representing stream positions.

The standard places no requirements upon the template parameter `StateT`. In this implementation `StateT` must be `DefaultConstructible`, `CopyConstructible` and `Assignable`. The standard only requires that `fpos` should contain a member of type `StateT`. In this implementation it also contains an offset stored as a signed integer.

## Parameters

<code>StateT</code>	Type passed to and returned from <code>state()</code> .
---------------------	---

Definition at line 112 of file postypes.h.

## 5.738.2 Constructor & Destructor Documentation

### 5.738.2.1 fpos()

```
template<typename _StateT>
std::fpos< _StateT >::fpos (
    streamoff __off ) [inline]
```

Construct position from offset.

Definition at line 133 of file postypes.h.

## 5.738.3 Member Function Documentation

### 5.738.3.1 operator streamoff()

```
template<typename _StateT>
std::fpos< _StateT >::operator streamoff ( ) const [inline]
```

Convert to streamoff.

Definition at line 137 of file postypes.h.

### 5.738.3.2 operator+()

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator+ (
    streamoff __off ) const [inline]
```

Add position and offset.

Definition at line 178 of file postypes.h.

### 5.738.3.3 operator+=()

```
template<typename _StateT>
fpos& std::fpos<_StateT>::operator+= (
    streamoff __off ) [inline]
```

Add offset to this position.

Definition at line 154 of file postypes.h.

### 5.738.3.4 operator-() [1/2]

```
template<typename _StateT>
fpos std::fpos<_StateT>::operator- (
    streamoff __off ) const [inline]
```

Subtract offset from position.

Definition at line 192 of file postypes.h.

### 5.738.3.5 operator-() [2/2]

```
template<typename _StateT>
streamoff std::fpos<_StateT>::operator- (
    const fpos<_StateT> & __other ) const [inline]
```

Subtract position to return offset.

Definition at line 205 of file postypes.h.

### 5.738.3.6 operator-=()

```
template<typename _StateT>
fpos& std::fpos<_StateT>::operator-= (
    streamoff __off ) [inline]
```

Subtract offset from this position.

Definition at line 165 of file postypes.h.



**5.738.3.7 state()** [1/2]

```
template<typename _StateT>
void std::fpos< _StateT >::state (
    _StateT __st ) [inline]
```

Remember the value of *st*.

Definition at line 141 of file postypes.h.

**5.738.3.8 state()** [2/2]

```
template<typename _StateT>
_StateT std::fpos< _StateT >::state ( ) const [inline]
```

Return the last set value of *st*.

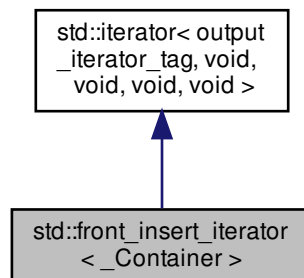
Definition at line 146 of file postypes.h.

The documentation for this class was generated from the following file:

- [postypes.h](#)

**5.739 std::front\_insert\_iterator< \_Container > Class Template Reference**

Inheritance diagram for `std::front_insert_iterator< _Container >`:



### Public Types

- typedef \_Container [container\\_type](#)
- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- [front\\_insert\\_iterator](#) (\_Container &\_\_x)
- [front\\_insert\\_iterator](#) & [operator\\*](#) ()
- [front\\_insert\\_iterator](#) & [operator++](#) ()
- [front\\_insert\\_iterator](#) [operator++](#) (int)
- [front\\_insert\\_iterator](#) & [operator=](#) (const typename \_Container::value\_type &\_\_value)
- [front\\_insert\\_iterator](#) & **[operator=](#)** (typename \_Container::value\_type &&\_\_value)

### Protected Attributes

- \_Container \* **[container](#)**

#### 5.739.1 Detailed Description

```
template<typename _Container>
class std::front_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 547 of file `bits/stl_iterator.h`.

#### 5.739.2 Member Typedef Documentation

##### 5.739.2.1 `container_type`

```
template<typename _Container >
typedef _Container std::front\_insert\_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

Definition at line 555 of file `bits/stl_iterator.h`.

#### 5.739.2.2 difference\_type

```
typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference\_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

#### 5.739.2.3 iterator\_category

```
typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void , void , void , void >↵
::iterator\_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

#### 5.739.2.4 pointer

```
typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

#### 5.739.2.5 reference

```
typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

#### 5.739.2.6 value\_type

```
typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::value\_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

### 5.739.3 Constructor & Destructor Documentation

#### 5.739.3.1 front\_insert\_iterator()

```
template<typename _Container >
std::front_insert_iterator< _Container >::front_insert_iterator (
    _Container & __x ) [inline], [explicit]
```

The only way to create this iterator is with a container.

Definition at line 558 of file bits/stl\_iterator.h.

### 5.739.4 Member Function Documentation

#### 5.739.4.1 operator\*()

```
template<typename _Container >
front_insert_iterator& std::front_insert_iterator< _Container >::operator* ( ) [inline]
```

Simply returns \*this.

Definition at line 597 of file bits/stl\_iterator.h.

#### 5.739.4.2 operator++() [1/2]

```
template<typename _Container >
front_insert_iterator& std::front_insert_iterator< _Container >::operator++ ( ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 602 of file bits/stl\_iterator.h.

#### 5.739.4.3 operator++() [2/2]

```
template<typename _Container >
front_insert_iterator std::front_insert_iterator< _Container >::operator++ (
    int ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 607 of file bits/stl\_iterator.h.

#### 5.739.4.4 operator=()

```
template<typename _Container >
front_insert_iterator& std::front_insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value ) [inline]
```

#### Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> .
----------------------	---

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

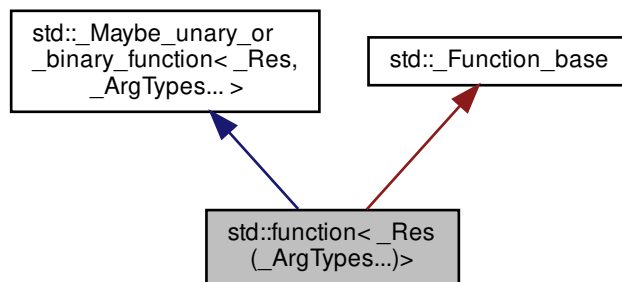
Definition at line 581 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

### 5.740 `std::function<_Res(_ArgTypes...)>` Class Template Reference

Inheritance diagram for `std::function<_Res(_ArgTypes...)>`:



#### Public Types

- `typedef _Res result_type`

## Public Member Functions

- [function](#) () noexcept
  - [function](#) (nullptr\_t) noexcept
  - [function](#) (const function &\_\_x)
  - [function](#) (function &&\_\_x) noexcept
  - template<typename \_Functor , typename = \_Requires<\_\_not\_<is\_same<\_Functor, function>>, void>, typename = \_Requires<\_Callable<\_Functor>, void>>>  
[function](#) (\_Functor)
  - [operator bool](#) () const noexcept
  - \_Res [operator\(\)](#) (\_ArgTypes... \_\_args) const
  - [function](#) & [operator=](#) (const [function](#) &\_\_x)
  - [function](#) & [operator=](#) (function &&\_\_x) noexcept
  - [function](#) & [operator=](#) (nullptr\_t) noexcept
  - template<typename \_Functor >  
\_Requires<\_Callable< typename [decay](#)<\_Functor>::type >, function & > [operator=](#) (\_Functor &&\_\_f)
  - template<typename \_Functor >  
[function](#) & [operator=](#) ([reference\\_wrapper](#)<\_Functor> \_\_f) noexcept
  - void [swap](#) (function &\_\_x) noexcept
  - const [type\\_info](#) & [target\\_type](#) () const noexcept
- 
- template<typename \_Functor >  
\_Functor \* [target](#) () noexcept
  - template<typename \_Functor >  
const \_Functor \* [target](#) () const noexcept

## Private Types

- typedef bool(\* **\_Manager\_type**) (\_Any\_data &, const \_Any\_data &, \_Manager\_operation)

## Private Member Functions

- bool **\_M\_empty** () const

## Private Attributes

- \_Any\_data **\_M\_functor**
- \_Manager\_type **\_M\_manager**

## Static Private Attributes

- static const std::size\_t **\_M\_max\_align**
- static const std::size\_t **\_M\_max\_size**

#### 5.740.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
class std::function< _Res(_ArgTypes...)>
```

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 370 of file std\_function.h.

#### 5.740.2 Constructor & Destructor Documentation

##### 5.740.2.1 function() [1/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function ( ) [inline], [noexcept]
```

Default construct creates an empty function call wrapper.

##### Postcondition

!(bool)\*this

Definition at line 395 of file std\_function.h.

##### 5.740.2.2 function() [2/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
    nullptr_t ) [inline], [noexcept]
```

Creates an empty function call wrapper.

##### Postcondition

!(bool)\*this


Definition at line 402 of file std\_function.h.

##### 5.740.2.3 function() [3/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
    const function< _Res(_ArgTypes...)> & __x )
```

Function copy constructor.

**Parameters**

	A function object with identical call signature.
<code>__x</code>	

**Postcondition**

```
bool(*this) == bool(__x)
```

The newly-created function contains a copy of the target of `__x` (if it has one).


Definition at line 653 of file `std_function.h`.

**5.740.2.4 function()** [4/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
    function< _Res(_ArgTypes...)> && __x ) [inline], [noexcept]
```

Function move constructor.

**Parameters**

	A function object rvalue with identical call signature.
<code>__x</code>	

The newly-created function contains the target of `__x` (if it has one).

Definition at line 422 of file `std_function.h`.

**5.740.2.5 function()** [5/5]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor , typename , typename >
std::function< _Res(_ArgTypes...)>::function (
    _Functor __f )
```

Builds a function that targets a copy of the incoming function object.



**Parameters**

<div><div>↩</div><div>↩</div><div>↩</div><div>↩</div><div><i>f</i></div></div>	A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res.
--	--

The newly-created function object will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If `__f` is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 667 of file `std_function.h`.

**5.740.3 Member Function Documentation****5.740.3.1 operator bool()**

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::operator bool ( ) const [inline], [explicit], [noexcept]
```

Determine if the function wrapper has a target.

**Returns**

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 563 of file `std_function.h`.

**5.740.3.2 operator()()**

```
template<typename _Res , typename... _ArgTypes>
_Res std::function< _Res(_ArgTypes...)>::operator() (
    _ArgTypes... __args ) const
```

Invokes the function targeted by `*this`.

**Returns**

the result of the target.

## Exceptions

<i>bad_function_call</i>	when <code>!(bool)*this</code>
--------------------------	--------------------------------

The function call operator invokes the target function object stored by `this`.

Definition at line 683 of file `std_function.h`.

## 5.740.3.3 operator=() [1/5]

```
template<typename _Res , typename... _ArgTypes>
function& std::function< _Res(_ArgTypes...)>::operator= (
    const function< _Res(_ArgTypes...)> & __x ) [inline]
```

Function assignment operator.

## Parameters

<code>__x</code>	A function with identical call signature.
------------------	---

## Postcondition

`(bool)*this == (bool)x`

## Returns

`*this`

The target of `__x` is copied to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 461 of file `std_function.h`.

## 5.740.3.4 operator=() [2/5]

```
template<typename _Res , typename... _ArgTypes>
function& std::function< _Res(_ArgTypes...)>::operator= (
    function< _Res(_ArgTypes...)> && __x ) [inline], [noexcept]
```

Function move-assignment operator.

**Parameters**

<code>__x</code>	A function rvalue with identical call signature.
------------------	--

**Returns**

`*this`

The target of `__x` is moved to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 479 of file `std_function.h`.

**5.740.3.5 operator=()** [3/5]

```
template<typename _Res , typename... _ArgTypes>
function& std::function< _Res(_ArgTypes...)>::operator= (
    nullptr_t ) [inline], [noexcept]
```

Function assignment to zero.

**Postcondition**

`!(bool)*this`

**Returns**

`*this`

The target of `*this` is deallocated, leaving it empty.

Definition at line 493 of file `std_function.h`.

**5.740.3.6 operator=()** [4/5]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
_requires<_Callable<typename decay<_Functor>::type>, function&> std::function< _Res(_ArgTypes...)>::operator= (
    _Functor && __f ) [inline]
```

Function assignment to a new target.

## Parameters

$\leftarrow$	A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$ <i>f</i>	

## Returns

\*this

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If `__f` is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 522 of file `std_function.h`.

## 5.740.3.7 operator=() [5/5]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
function& std::function< _Res(_ArgTypes...)>::operator= (
    reference_wrapper< _Functor > __f ) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 531 of file `std_function.h`.

## 5.740.3.8 swap()

```
template<typename _Res , typename... _ArgTypes>
void std::function< _Res(_ArgTypes...)>::swap (
    function< _Res(_ArgTypes...)> & __x ) [inline], [noexcept]
```

Swap the targets of two function objects.

## Parameters

$\_ \leftarrow$	A function with identical call signature.
$\_x$	

Swap the targets of `this` function object and `__f`. This function will not throw an exception.

Definition at line 546 of file `std_function.h`.

#### 5.740.3.9 `target()` [1/2]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
_Functor * std::function< _Res(_ArgTypes...)>::target ( ) [noexcept]
```

Access the stored target function object.

##### Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a NULL pointer.

This function does not throw exceptions.

Definition at line 710 of file `std_function.h`.

#### 5.740.3.10 `target()` [2/2]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
const _Functor * std::function< _Res(_ArgTypes...)>::target ( ) const [noexcept]
```

Access the stored target function object.

##### Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a NULL pointer.

This function does not throw exceptions.

Definition at line 721 of file `std_function.h`.

5.740.3.11 `target_type()`

```
template<typename _Res , typename... _ArgTypes>
const type\_info & std::function<_Res(_ArgTypes...)>::target_type ( ) const [noexcept]
```

Determine the type of the target of this function object wrapper.

**Returns**

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

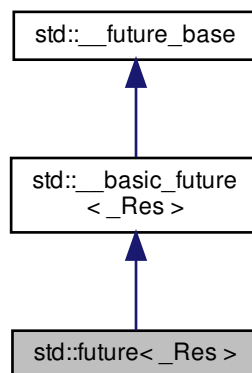
Definition at line 694 of file `std_function.h`.

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

5.741 `std::future<_Res>` Class Template Reference

Inheritance diagram for `std::future<_Res>`:

**Public Types**

- `template<typename _Res>`  
`using \_Ptr = unique\_ptr<_Res, _Result_base::Deleter>`
- `using _State_base = _State_baseV2`

## Public Member Functions

- `future` (`future` &&\_\_uf) noexcept
- `future` (const `future` &)=delete
- `_Res` `get` ()
- `future` & `operator=` (const `future` &)=delete
- `future` & `operator=` (`future` &&\_\_fut) noexcept
- `shared_future`< `_Res` > `share` () noexcept
- bool `valid` () const noexcept
- void `wait` () const
- template<typename `_Rep` , typename `_Period` >  
`future_status` `wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rel) const
- template<typename `_Clock` , typename `_Duration` >  
`future_status` `wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs) const

## Static Public Member Functions

- template<typename `_Res` , typename `_Allocator` >  
static `_Ptr`< `_Result_alloc`< `_Res`, `_Allocator` > > `_S_allocate_result` (const `_Allocator` &\_\_a)
- template<typename `_Res` , typename `_Tp` >  
static `_Ptr`< `_Result`< `_Res` > > `_S_allocate_result` (const `std::allocator`< `_Tp` > &\_\_a)
- template<typename `_BoundFn` >  
static `std::shared_ptr`< `_State_base` > `_S_make_async_state` (`_BoundFn` &&\_\_fn)
- template<typename `_BoundFn` >  
static `std::shared_ptr`< `_State_base` > `_S_make_deferred_state` (`_BoundFn` &&\_\_fn)
- template<typename `_Res_ptr` , typename `_BoundFn` >  
static `_Task_setter`< `_Res_ptr`, `_BoundFn` > `_S_task_setter` (`_Res_ptr` &\_\_ptr, `_BoundFn` &\_\_call)

## Protected Types

- typedef `__future_base::Result`< `_Res` > & `__result_type`

## Protected Member Functions

- `__result_type` `_M_get_result` () const
- void `_M_swap` (`__basic_future` &\_\_that) noexcept

## Friends

- template<typename `_Fn` , typename... `_Args`>  
`future`< `__async_result_of`< `_Fn`, `_Args`... > > `async` (`launch`, `_Fn` &&, `_Args` &&...)
- template<typename >  
class `packaged_task`
- class `promise`< `_Res` >

### 5.741.1 Detailed Description

```
template<typename _Res>  
class std::future< _Res >
```

Primary template for future.

Definition at line 125 of file future.

### 5.741.2 Member Typedef Documentation

#### 5.741.2.1 \_Ptr

```
template<typename _Res >  
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A unique\_ptr for result objects.

Definition at line 223 of file future.

### 5.741.3 Constructor & Destructor Documentation

#### 5.741.3.1 future()

```
template<typename _Res >  
std::future< _Res >::future (   
    future< _Res > && __uf ) [inline], [noexcept]
```

Move constructor.

Definition at line 779 of file future.

### 5.741.4 Member Function Documentation

#### 5.741.4.1 \_M\_get\_result()

```
template<typename _Res>  
__result_type std::__basic_future< _Res >::_M_get_result ( ) const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 714 of file future.



### 5.741.4.2 `get()`

```
template<typename _Res >
_Res std::future< _Res >::get ( ) [inline]
```

Retrieving the value.

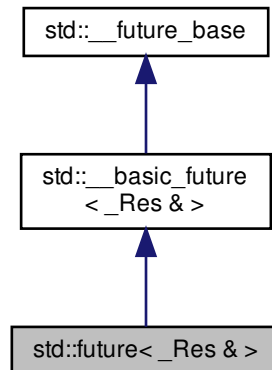
Definition at line 793 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.742 `std::future< _Res & >` Class Template Reference

Inheritance diagram for `std::future< _Res & >`:



### Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr< _Res, _Result_base::_Deleter >`
- `using _State_base = _State_baseV2`

### Public Member Functions

- `future (future &&__uf) noexcept`
- `future (const future &)=delete`
- `_Res & get ()`
- `future & operator= (const future &)=delete`
- `future & operator= (future &&__fut) noexcept`
- `shared\_future< _Res & > share () noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `future\_status wait_for (const chrono::duration< _Rep, _Period > &__rel) const`
- `future\_status wait_until (const chrono::time\_point< _Clock, _Duration > &__abs) const`

### Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static _Ptr< _Result_alloc< _Res, _Allocator > > _S_allocate_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static _Ptr< _Result< _Res > > _S_allocate_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_async_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_deferred_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static _Task_setter< _Res_ptr, _BoundFn > _S_task_setter (_Res_ptr &__ptr, _BoundFn &__call)`

### Protected Types

- `typedef __future_base::_Result< _Res & > & __result_type`

### Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

### Friends

- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch, _Fn &&, _Args &&...)`
- `template<typename >`  
`class packaged_task`
- `class promise< _Res & >`

#### 5.742.1 Detailed Description

```
template<typename _Res>
class std::future< _Res & >
```

Partial specialization for future<R&>

Definition at line 804 of file future.

#### 5.742.2 Member Typedef Documentation

#### 5.742.2.1 `_Ptr`

```
template<typename _Res >
using std::\_\_future\_base::\_Ptr = unique\_ptr<_Res, \_Result\_base::\_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file `future`.

### 5.742.3 Constructor & Destructor Documentation

#### 5.742.3.1 `future()`

```
template<typename _Res >
std::future< _Res & >::future (
    future< _Res & > && \_\_uf ) [inline], [noexcept]
```

Move constructor.

Definition at line 822 of file `future`.

### 5.742.4 Member Function Documentation

#### 5.742.4.1 `_M_get_result()`

```
\_\_result\_type std::\_\_basic\_future< _Res & >::\_M\_get\_result ( ) const [inline], [protected],
[inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 714 of file `future`.

#### 5.742.4.2 `get()`

```
template<typename _Res >
_Res& std::future< _Res & >::get ( ) [inline]
```

Retrieving the value.

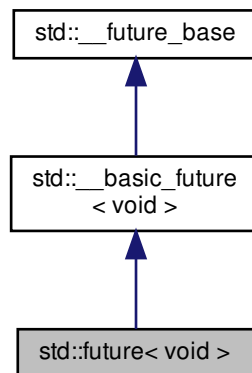
Definition at line 836 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

5.743 `std::future< void >` Class Template Reference

Inheritance diagram for `std::future< void >`:



## Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr< _Res, _Result_base::_Deleter >`
- `using _State_base = _State_baseV2`

## Public Member Functions

- `future (future &&__uf) noexcept`
- `future (const future &)=delete`
- `void get ()`
- `future & operator= (const future &)=delete`
- `future & operator= (future &&__fut) noexcept`
- `shared\_future< void > share () noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `future\_status wait_for (const chrono::duration< _Rep, _Period > &__rel) const`
- `future\_status wait_until (const chrono::time\_point< _Clock, _Duration > &__abs) const`

### Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static _Ptr< _Result_alloc< _Res, _Allocator > > _S_allocate_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static _Ptr< _Result< _Res > > _S_allocate_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_async_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_deferred_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static _Task_setter< _Res_ptr, _BoundFn > _S_task_setter (_Res_ptr &__ptr, _BoundFn &__call)`

### Protected Types

- `typedef __future_base::_Result< void > & __result_type`

### Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

### Friends

- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch, _Fn &&, _Args &&...)`
- `template<typename >`  
`class packaged_task`
- `class promise< void >`

#### 5.743.1 Detailed Description

```
template<>
class std::future< void >
```

Explicit specialization for `future<void>`

Definition at line 847 of file `future`.

#### 5.743.2 Member Typedef Documentation

#### 5.743.2.1 `_Ptr`

```
template<typename _Res >  
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file `future`.

### 5.743.3 Constructor & Destructor Documentation

#### 5.743.3.1 `future()`

```
std::future< void >::future (  
    future< void > && __uf ) [inline], [noexcept]
```

Move constructor.

Definition at line 865 of file `future`.

### 5.743.4 Member Function Documentation

#### 5.743.4.1 `_M_get_result()`

```
__result_type std::__basic_future< void >::_M_get_result ( ) const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 714 of file `future`.

#### 5.743.4.2 `get()`

```
void std::future< void >::get ( ) [inline]
```

Retrieving the value.

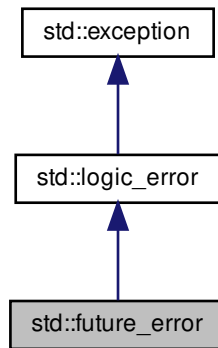
Definition at line 879 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 5.744 `std::future_error` Class Reference

Inheritance diagram for `std::future_error`:



### Public Member Functions

- **`future_error`** (`future_errc` \_\_errc)
- const `error_code` & **`code`** () const noexcept
- virtual const char \* `what` () const noexcept

### Friends

- void **`__throw_future_error`** (int)

### 5.744.1 Detailed Description

Exception type thrown by futures.

Definition at line 96 of file future.

### 5.744.2 Member Function Documentation

5.744.2.1 `what()`

```
virtual const char* std::future_error::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic\\_error](#).

The documentation for this class was generated from the following file:

- [future](#)

5.745 `std::gamma_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [gamma\\_distribution](#) (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- **[gamma\\_distribution](#)** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **[generate](#)** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **[generate](#)** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >  
void **[generate](#)** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_RealType` [alpha](#) () const
- `_RealType` [beta](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()



## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::gamma_distribution< _RealType1 > &__x)`
- `bool operator== (const gamma_distribution &__d1, const gamma_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::gamma_distribution< _RealType1 > &__x)`

### 5.745.1 Detailed Description

```
template<typename _RealType = double>
class std::gamma_distribution< _RealType >
```

A gamma continuous distribution for random numbers.

The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2352 of file random.h.

### 5.745.2 Member Typedef Documentation

#### 5.745.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::gamma_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2355 of file random.h.

### 5.745.3 Constructor & Destructor Documentation

#### 5.745.3.1 gamma\_distribution()

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution (
    _RealType __alpha_val = _RealType(1),
    _RealType __beta_val = _RealType(1) ) [inline], [explicit]
```

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

Definition at line 2409 of file random.h.

#### 5.745.4 Member Function Documentation

##### 5.745.4.1 alpha()

```
template<typename _RealType = double>
_RealType std::gamma_distribution<_RealType>::alpha ( ) const [inline]
```

Returns the  $\alpha$  of the distribution.

Definition at line 2430 of file random.h.

##### 5.745.4.2 beta()

```
template<typename _RealType = double>
_RealType std::gamma_distribution<_RealType>::beta ( ) const [inline]
```

Returns the  $\beta$  of the distribution.

Definition at line 2437 of file random.h.

##### 5.745.4.3 max()

```
template<typename _RealType = double>
result_type std::gamma_distribution<_RealType>::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2466 of file random.h.

##### 5.745.4.4 min()

```
template<typename _RealType = double>
result_type std::gamma_distribution<_RealType>::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2459 of file random.h.

**5.745.4.5 operator() [1/2]**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::gamma_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 2474 of file random.h.

Referenced by std::gamma\_distribution< result\_type >::operator()().

**5.745.4.6 operator() [2/2]**

```
template<typename _RealType >
template<typename _UniformRandomNumberGenerator >
gamma_distribution< _RealType >::result_type std::gamma_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param )
```

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2344 of file bits/random.tcc.

**5.745.4.7 param() [1/2]**

```
template<typename _RealType = double>
param_type std::gamma_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2444 of file random.h.

Referenced by std::chi\_squared\_distribution< \_RealType >::param().

**5.745.4.8 param() [2/2]**

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2452 of file random.h.

## 5.745.4.9 reset()

```
template<typename _RealType = double>
void std::gamma_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 2423 of file random.h.

Referenced by `std::chi_squared_distribution<_RealType>::reset()`, `std::fisher_f_distribution<_RealType>::reset()`, `std::student_t_distribution<_RealType>::reset()`, and `std::negative_binomial_distribution<_IntType>::reset()`.

## 5.745.5 Friends And Related Function Documentation

## 5.745.5.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::gamma_distribution<_RealType1> & __x ) [friend]
```

Inserts a `gamma_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>gamma_distribution</code> random number distribution.

## Returns

The output stream with the state of `__x` inserted or in an error state.

### 5.745.5.2 operator==

```
template<typename _RealType = double>
bool operator== (
    const gamma\_distribution< _RealType > & __d1,
    const gamma\_distribution< _RealType > & __d2 ) [friend]
```

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2510 of file random.h.

### 5.745.5.3 operator>>

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic\_istream<_CharT, _Traits>& operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::gamma\_distribution< _RealType1 > & __x ) [friend]
```

Extracts a [gamma\\_distribution](#) random number distribution `__x` from the input stream `__is`.

#### Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <a href="#">gamma_distribution</a> random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.746 [std::gamma\\_distribution](#)< \_RealType >::param\_type Struct Reference

#### Public Types

- typedef [gamma\\_distribution](#)< \_RealType > **distribution\_type**

## Public Member Functions

- `param_type` (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- `_RealType alpha` () const
- `_RealType beta` () const

## Friends

- class `gamma_distribution<_RealType>`
- bool `operator!=` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 5.746.1 Detailed Description

```
template<typename _RealType = double>
struct std::gamma_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2362 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.747 `std::geometric_distribution<_IntType>` Class Template Reference

## Classes

- struct `param_type`

## Public Types

- typedef `_IntType result_type`

## Public Member Functions

- **geometric\_distribution** (double \_\_p=0.5)
- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate (result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool [operator==](#) (const [geometric\\_distribution](#) &\_\_d1, const [geometric\\_distribution](#) &\_\_d2)

### 5.747.1 Detailed Description

```
template<typename _IntType = int>
class std::geometric_distribution< _IntType >
```

A discrete geometric random number distribution.

The formula for the geometric probability density function is  $p(i|p) = p(1 - p)^i$  where  $p$  is the parameter of the distribution.

Definition at line 3898 of file random.h.

### 5.747.2 Member Typedef Documentation

#### 5.747.2.1 [result\\_type](#)

```
template<typename _IntType = int>
typedef _IntType std::geometric\_distribution< _IntType >::result\_type
```

The type of the range of the distribution.

Definition at line 3901 of file random.h.

### 5.747.3 Member Function Documentation

#### 5.747.3.1 max()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3995 of file random.h.

References std::numeric\_limits< \_Tp >::max().

#### 5.747.3.2 min()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3988 of file random.h.

#### 5.747.3.3 operator>()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::geometric_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 4003 of file random.h.

#### 5.747.3.4 p()

```
template<typename _IntType = int>
double std::geometric_distribution< _IntType >::p ( ) const [inline]
```

Returns the distribution parameter p.

Definition at line 3966 of file random.h.

Referenced by std::operator<<().



**5.747.3.5 param()** [1/2]

```
template<typename _IntType = int>
param_type std::geometric_distribution< _IntType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3973 of file random.h.

Referenced by std::operator>>().

**5.747.3.6 param()** [2/2]

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3981 of file random.h.

**5.747.3.7 reset()**

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::reset ( ) [inline]
```

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 3960 of file random.h.

**5.747.4 Friends And Related Function Documentation**

5.747.4.1 `operator==`

```
template<typename _IntType = int>
bool operator== (
    const geometric\_distribution< _IntType > & __d1,
    const geometric\_distribution< _IntType > & __d2 ) [friend]
```

Return true if two geometric distributions have the same parameters.

Definition at line 4038 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.748 `std::geometric_distribution< _IntType >::param_type` Struct Reference

## Public Types

- typedef [geometric\\_distribution](#)< \_IntType > **distribution\_type**

## Public Member Functions

- **param\_type** (double \_\_p=0.5)
- double **p** () const

## Friends

- class [geometric\\_distribution](#)< \_IntType >
- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.748.1 Detailed Description

```
template<typename _IntType = int>
struct std::geometric_distribution< _IntType >::param_type
```

Parameter type.

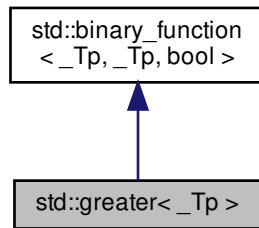
Definition at line 3908 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.749 `std::greater<_Tp>` Struct Template Reference

Inheritance diagram for `std::greater<_Tp>`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.749.1 Detailed Description

```
template<typename _Tp>  
struct std::greater<_Tp>
```

One of the [comparison functors](#).

Definition at line 337 of file `stl_function.h`.

#### 5.749.2 Member Typedef Documentation

5.749.2.1 `first_argument_type`

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

5.749.2.2 `result_type`

```
typedef bool std::binary\_function< _Tp , _Tp , bool >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

5.749.2.3 `second_argument_type`

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

5.750 `std::greater< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`__t`, `_Up` &&`__u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`__t`) > [std::forward](#)< `_Up` >(`__u`))) -> decltype([std::forward](#)< `_Tp` >(`__t`) > [std::forward](#)< `_Up` >(`__u`))
- template<typename `_Tp`, typename `_Up`>  
constexpr bool **operator()** (`_Tp` \*`__t`, `_Up` \*`__u`) const noexcept

### 5.750.1 Detailed Description

```
template<>
struct std::greater< void >
```

One of the [comparison functors](#).

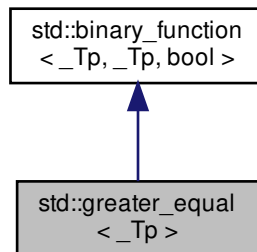
Definition at line 492 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.751 `std::greater_equal<_Tp>` Struct Template Reference

Inheritance diagram for `std::greater_equal<_Tp>`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR bool operator()(const _Tp &__x, const _Tp &__y) const`

### 5.751.1 Detailed Description

```
template<typename _Tp>
struct std::greater_equal<_Tp>
```

One of the [comparison functors](#).

Definition at line 343 of file stl\_function.h.

### 5.751.2 Member Typedef Documentation

#### 5.751.2.1 first\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.751.2.2 result\_type

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.751.2.3 second\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.752 `std::greater_equal< void >` Struct Template Reference

### Public Types

- typedef `__is_transparent` **`is_transparent`**

### Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **`operator()`** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`_t`) >=[std::forward](#)< `_Up` >(`_u`))) -> decltype([std::forward](#)< `_Tp` >(`_t`) >=[std::forward](#)< `_Up` >(`_u`))
- template<typename `_Tp` , typename `_Up` >  
constexpr bool **`operator()`** (`_Tp` \*`_t`, `_Up` \*`_u`) const noexcept

### 5.752.1 Detailed Description

```
template<>
struct std::greater_equal< void >
```

One of the [comparison functors](#).

Definition at line 616 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.753 `std::gslice` Class Reference

### Public Member Functions

- [gslice](#) ()
- [gslice](#) (size\_t `__o`, const [valarray](#)< size\_t > &`__l`, const [valarray](#)< size\_t > &`__s`)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & **`operator=`** (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- size\_t [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

### Friends

- template<typename `_Tp` >  
class **`valarray`**

## 5.753.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 64 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

5.754 `std::gslice_array<_Tp>` Class Template Reference

## Public Types

- `typedef _Tp value_type`

## Public Member Functions

- [gslice\\_array](#) (const [gslice\\_array](#) &)
- `void operator%=(const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator%=(const _Expr<_Dom, _Tp> &) const`
- `void operator&=(const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator&=(const _Expr<_Dom, _Tp> &) const`
- `void operator*=(const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator*=(const _Expr<_Dom, _Tp> &) const`
- `void operator+=(const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator+=(const _Expr<_Dom, _Tp> &) const`
- `void operator-=(const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator-=(const _Expr<_Dom, _Tp> &) const`
- `void operator/=(const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator/=(const _Expr<_Dom, _Tp> &) const`
- `void operator<=<= (const valarray<_Tp> &) const`
- `template<class _Dom>`  
`void operator<=<= (const _Expr<_Dom, _Tp> &) const`



- [gslice\\_array](#) & [operator=](#) (const [gslice\\_array](#) &)
- void [operator=](#) (const [valarray](#)< \_Tp > &) const
- void [operator=](#) (const \_Tp &) const
- template<class \_Dom >  
void [operator=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator>>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator>>=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator^=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator^=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator|=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator|=](#) (const \_Expr< \_Dom, \_Tp > &) const

#### Friends

- class [valarray](#)< \_Tp >

#### 5.754.1 Detailed Description

```
template<typename _Tp>  
class std::gslice_array< _Tp >
```

Reference to multi-dimensional subset of an array.

A [gslice\\_array](#) is a reference to the actual elements of an array specified by a [gslice](#). The way to get a [gslice\\_array](#) is to call [operator\[\]](#)([gslice](#)) on a [valarray](#). The returned [gslice\\_array](#) then permits carrying operations out on the referenced subset of elements in the original [valarray](#). For example, [operator+=\(valarray\)](#) will add values to the subset of elements in the underlying [valarray](#) this [gslice\\_array](#) refers to.

#### Parameters

<i>Tp</i>	Element type.
-----------	---------------

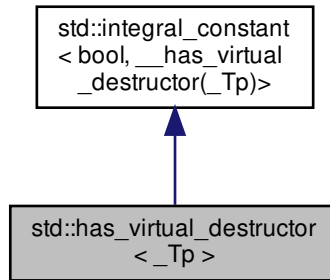
Definition at line 82 of file [valarray](#).

The documentation for this class was generated from the following files:

- [valarray](#)
- [gslice\\_array.h](#)

## 5.755 std::has\_virtual\_destructor&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::has\_virtual\_destructor< \_Tp >:

**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

**Static Public Attributes**

- static constexpr bool **value**

## 5.755.1 Detailed Description

```
template<typename _Tp>
struct std::has_virtual_destructor< _Tp >
```

has\_virtual\_destructor

Definition at line 1230 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.756 `std::hash< _Tp >` Struct Template Reference

Inherits `std::__hash_enum< _Tp, bool >`.

#### 5.756.1 Detailed Description

```
template<typename _Tp>
struct std::hash< _Tp >
```

Primary class template hash.

Primary class template hash, usable for enum types only.

Definition at line 142 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

### 5.757 `std::hash< __debug::bitset< _Nb > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (const __debug::bitset< _Nb > &__b) const` `noexcept`

#### 5.757.1 Detailed Description

```
template<size_t _Nb>
struct std::hash< __debug::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 416 of file `debug/bitset`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

## 5.758 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (const __debug::vector< bool, _Alloc > &__b) const` `noexcept`

#### 5.758.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< __debug::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 778 of file `debug/vector`.

The documentation for this struct was generated from the following file:

- [debug/vector](#)

## 5.759 `std::hash< __gnu_cxx::__u16vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (const __gnu_cxx::__u16vstring &__s) const` `noexcept`

#### 5.759.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__u16vstring >
```

std::hash specialization for \_\_u16vstring.

Definition at line 2939 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 5.760 std::hash< \_\_gnu\_cxx::\_\_u32vstring > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

##### Public Member Functions

- size\_t **operator()** (const [\\_\\_gnu\\_cxx::\\_\\_u32vstring](#) &\_\_s) const noexcept

#### 5.760.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__u32vstring >
```

std::hash specialization for \_\_u32vstring.

Definition at line 2950 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 5.761 std::hash< \_\_gnu\_cxx::\_\_vstring > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- size\_t **operator()** (const [\\_\\_gnu\\_cxx::\\_\\_vstring](#) &\_\_s) const noexcept

#### 5.761.1 Detailed Description

template<>

struct std::hash< \_\_gnu\_cxx::\_\_vstring >

std::hash specialization for \_\_vstring.

Definition at line 2915 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

## 5.762 std::hash< \_\_gnu\_cxx::\_\_wvstring > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- size\_t **operator()** (const [\\_\\_gnu\\_cxx::\\_\\_wvstring](#) &\_\_s) const noexcept

#### 5.762.1 Detailed Description

template<>

struct std::hash< \_\_gnu\_cxx::\_\_wvstring >

std::hash specialization for \_\_wvstring.

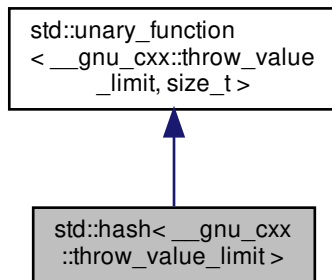
Definition at line 2926 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

### 5.763 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



#### Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

#### Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_limit` &`__val`) const

#### 5.763.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::throw_value_limit >
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 950 of file `throw_allocator.h`.

#### 5.763.2 Member Typedef Documentation

5.763.2.1 `argument_type`

```
typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit , size_t  
>::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

5.763.2.2 `result_type`

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_limit , size_t >::result_type [inherited]
```

`result_type` is the return type

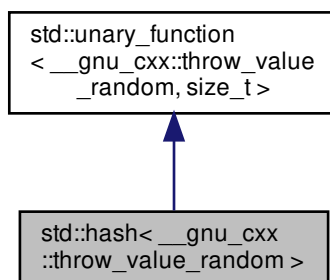
Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

5.764 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



## Public Types

- typedef [\\_\\_gnu\\_cxx::throw\\_value\\_random](#) `argument_type`
- typedef `size_t` `result_type`



## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::throw\\_value\\_random](#) &\_\_val) const

### 5.764.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::throw_value_random >
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_random`.

Definition at line 965 of file `throw_allocator.h`.

### 5.764.2 Member Typedef Documentation

#### 5.764.2.1 `argument_type`

```
typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random , size_t
>::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.764.2.2 `result_type`

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.765 `std::hash< __profile::bitset< _Nb > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- size\_t **operator()** (const [\\_\\_profile::bitset](#)< \_Nb > &\_\_b) const noexcept

#### 5.765.1 Detailed Description

```
template<size_t _Nb>
struct std::hash< __profile::bitset< _Nb > >
```

std::hash specialization for bitset.

Definition at line 234 of file profile/bitset.

The documentation for this struct was generated from the following file:

- [profile/bitset](#)

## 5.766 std::hash< \_\_profile::vector< bool, \_Alloc > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- size\_t **operator()** (const \_\_profile::vector< bool, \_Alloc > &\_\_b) const noexcept

#### 5.766.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< __profile::vector< bool, _Alloc > >
```

std::hash specialization for vector<bool>.

Definition at line 559 of file profile/vector.

The documentation for this struct was generated from the following file:

- [profile/vector](#)

### 5.767 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const` **noexcept**

#### 5.767.1 Detailed Description

```
template<typename _Tp, _Lock_policy _Lp>
struct std::hash< __shared_ptr< _Tp, _Lp > >
```

`std::hash` specialization for `__shared_ptr`.

Definition at line 1853 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

### 5.768 `std::hash< _Tp * >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (_Tp *__p) const` **noexcept**

## 5.768.1 Detailed Description

```
template<typename _Tp>
struct std::hash< _Tp * >
```

Partial specializations for pointer types.

Definition at line 106 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

5.769 `std::hash< bool >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Result result_type _GLIBCXX17_DEPRECATED`
- `typedef _Arg argument_type _GLIBCXX17_DEPRECATED`

## Public Member Functions

- `size_t operator() (bool __val) const noexcept`

## 5.769.1 Detailed Description

```
template<>
struct std::hash< bool >
```

Explicit specialization for `bool`.

Definition at line 124 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

5.770 `std::hash< char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- size\_t **operator()** (char \_\_val) const noexcept

##### 5.770.1 Detailed Description

```
template<>
struct std::hash< char >
```

Explicit specialization for char.

Definition at line 127 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

##### 5.771 std::hash< char16\_t > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- size\_t **operator()** (char16\_t \_\_val) const noexcept

##### 5.771.1 Detailed Description

```
template<>
struct std::hash< char16_t >
```

Explicit specialization for char16\_t.

Definition at line 139 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.772 `std::hash< char32_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (char32_t __val) const` **noexcept**

#### 5.772.1 Detailed Description

```
template<>
struct std::hash< char32_t >
```

Explicit specialization for `char32_t`.

Definition at line 142 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.773 `std::hash< double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (double __val) const` **noexcept**

#### 5.773.1 Detailed Description

```
template<>
struct std::hash< double >
```

Specialization for double.

Definition at line 238 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 5.774 std::hash< error\_code > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

##### Public Member Functions

- size\_t **operator() (const [error\\_code](#) &\_\_e) const** noexcept

#### 5.774.1 Detailed Description

```
template<>
struct std::hash< error_code >
```

std::hash specialization for error\_code.

Definition at line 386 of file system\_error.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 5.775 std::hash< experimental::shared\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- size\_t **operator()** (const experimental::shared\_ptr< \_Tp > &\_\_s) const noexcept

##### 5.775.1 Detailed Description

```
template<typename _Tp>
struct std::hash< experimental::shared_ptr< _Tp > >
```

std::hash specialization for shared\_ptr.

Definition at line 665 of file experimental/bits/shared\_ptr.h.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

## 5.776 std::hash< float > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- size\_t **operator()** (float \_\_val) const noexcept

##### 5.776.1 Detailed Description

```
template<>
struct std::hash< float >
```

Specialization for float.

Definition at line 226 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)



### 5.777 `std::hash< int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (int __val) const` **noexcept**

#### 5.777.1 Detailed Description

```
template<>
struct std::hash< int >
```

Explicit specialization for `int`.

Definition at line 148 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.778 `std::hash< long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (long __val) const` **noexcept**

## 5.778.1 Detailed Description

```
template<>
struct std::hash< long >
```

Explicit specialization for long.

Definition at line 151 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

5.779 `std::hash< long double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Result result_type _GLIBCXX17_DEPRECATED`
- `typedef _Arg argument_type _GLIBCXX17_DEPRECATED`

## Public Member Functions

- `size_t operator() (long double __val) const noexcept`

## 5.779.1 Detailed Description

```
template<>
struct std::hash< long double >
```

Specialization for long double.

Definition at line 250 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

5.780 `std::hash< long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Result` `result_type` **`_GLIBCXX17_DEPRECATED`**
- typedef `_Arg` `argument_type` **`_GLIBCXX17_DEPRECATED`**

#### Public Member Functions

- `size_t` **`operator()`** (`long long` `__val`) `const` `noexcept`

##### 5.780.1 Detailed Description

```
template<>
struct std::hash< long long >
```

Explicit specialization for long long.

Definition at line 154 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

##### 5.781 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Result` `result_type` **`_GLIBCXX17_DEPRECATED`**
- typedef `_Arg` `argument_type` **`_GLIBCXX17_DEPRECATED`**

#### Public Member Functions

- `size_t` **`operator()`** (`const` [shared\\_ptr](#)< `_Tp` > &`__s`) `const` `noexcept`

##### 5.781.1 Detailed Description

```
template<typename _Tp>
struct std::hash< shared_ptr< _Tp > >
```

`std::hash` specialization for `shared_ptr`.

Definition at line 727 of file `bits/shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.782 `std::hash< short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (short __val) const` **noexcept**

#### 5.782.1 Detailed Description

```
template<>
struct std::hash< short >
```

Explicit specialization for `short`.

Definition at line 145 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.783 `std::hash< signed char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (signed char __val) const` **noexcept**

#### 5.783.1 Detailed Description

```
template<>
struct std::hash< signed char >
```

Explicit specialization for signed char.

Definition at line 130 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 5.784 std::hash< string > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

##### Public Member Functions

- size\_t **operator() (**const [string](#) &\_\_s) const noexcept

#### 5.784.1 Detailed Description

```
template<>
struct std::hash< string >
```

std::hash specialization for string.

Definition at line 6641 of file basic\_string.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

#### 5.785 std::hash< thread::id > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- `typedef _Result result_type` `_GLIBCXX17_DEPRECATED`
- `typedef _Arg argument_type` `_GLIBCXX17_DEPRECATED`

#### Public Member Functions

- `size_t operator()` (const [thread::id](#) &\_\_id) const noexcept

##### 5.785.1 Detailed Description

```
template<>
struct std::hash< thread::id >
```

`std::hash` specialization for `thread::id`.

Definition at line 313 of file `thread`.

The documentation for this struct was generated from the following file:

- [thread](#)

## 5.786 `std::hash< type_index >` Struct Template Reference

#### Public Types

- `typedef` [type\\_index](#) `argument_type`
- `typedef size_t` `result_type`

#### Public Member Functions

- `size_t operator()` (const [type\\_index](#) &\_\_ti) const noexcept

##### 5.786.1 Detailed Description

```
template<>
struct std::hash< type_index >
```

`std::hash` specialization for `type_index`.

Definition at line 97 of file `typeindex`.

The documentation for this struct was generated from the following file:

- [typeindex](#)

### 5.787 `std::hash< u16string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (const u16string &__s) const` `noexcept`

#### 5.787.1 Detailed Description

```
template<>
struct std::hash< u16string >
```

`std::hash` specialization for `u16string`.

Definition at line 6674 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

### 5.788 `std::hash< u32string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (const u32string &__s) const` `noexcept`

## 5.788.1 Detailed Description

```
template<>
struct std::hash< u32string >
```

`std::hash` specialization for `u32string`.

Definition at line 6689 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

5.789 `std::hash< unique_ptr< _Tp, _Dp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`, and `std::__poison_hash< _Tp, typename >`.

## Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

## Public Member Functions

- `size_t operator() (const unique\_ptr< _Tp, _Dp > &__u) const` noexcept

## Static Private Attributes

- `static constexpr bool __enable_hash_call`

## 5.789.1 Detailed Description

```
template<typename _Tp, typename _Dp>
struct std::hash< unique_ptr< _Tp, _Dp > >
```

`std::hash` specialization for `unique_ptr`.

Definition at line 803 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)



### 5.790 `std::hash< unsigned char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (unsigned char __val) const` **noexcept**

#### 5.790.1 Detailed Description

```
template<>
struct std::hash< unsigned char >
```

Explicit specialization for unsigned char.

Definition at line 133 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 5.791 `std::hash< unsigned int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- `size_t operator() (unsigned int __val) const` **noexcept**

## 5.791.1 Detailed Description

```
template<>
struct std::hash< unsigned int >
```

Explicit specialization for unsigned int.

Definition at line 160 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

5.792 `std::hash< unsigned long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

## Public Member Functions

- `size_t operator() (unsigned long __val) const` **noexcept**

## 5.792.1 Detailed Description

```
template<>
struct std::hash< unsigned long >
```

Explicit specialization for unsigned long.

Definition at line 163 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

5.793 `std::hash< unsigned long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Result` `result_type` `_GLIBCXX17_DEPRECATED`
- typedef `_Arg` `argument_type` `_GLIBCXX17_DEPRECATED`

#### Public Member Functions

- `size_t` **operator()** (unsigned long long `__val`) const noexcept

##### 5.793.1 Detailed Description

```
template<>
struct std::hash< unsigned long long >
```

Explicit specialization for unsigned long long.

Definition at line 166 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

##### 5.794 `std::hash< unsigned short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Result` `result_type` `_GLIBCXX17_DEPRECATED`
- typedef `_Arg` `argument_type` `_GLIBCXX17_DEPRECATED`

#### Public Member Functions

- `size_t` **operator()** (unsigned short `__val`) const noexcept

##### 5.794.1 Detailed Description

```
template<>
struct std::hash< unsigned short >
```

Explicit specialization for unsigned short.

Definition at line 157 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.795 `std::hash< wchar_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (wchar_t __val) const` **noexcept**

#### 5.795.1 Detailed Description

```
template<>
struct std::hash< wchar_t >
```

Explicit specialization for `wchar_t`.

Definition at line 136 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.796 `std::hash< wstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- `typedef _Result result_type` **\_GLIBCXX17\_DEPRECATED**
- `typedef _Arg argument_type` **\_GLIBCXX17\_DEPRECATED**

### Public Member Functions

- `size_t operator() (const wstring &__s) const` **noexcept**

### 5.796.1 Detailed Description

```
template<>
struct std::hash< wstring >
```

std::hash specialization for wstring.

Definition at line 6656 of file basic\_string.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

### 5.797 std::hash<::bitset< \_Nb > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Result result\_type **\_GLIBCXX17\_DEPRECATED**
- typedef \_Arg argument\_type **\_GLIBCXX17\_DEPRECATED**

#### Public Member Functions

- size\_t **operator()** (const ::[bitset](#)< \_Nb > &\_\_b) const noexcept

### 5.797.1 Detailed Description

```
template<size_t _Nb>
struct std::hash<::bitset< _Nb > >
```

std::hash specialization for bitset.

Definition at line 1563 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

### 5.798 std::hash<::vector< bool, \_Alloc > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- `typedef _Result result_type` `_GLIBCXX17_DEPRECATED`
- `typedef _Arg argument_type` `_GLIBCXX17_DEPRECATED`

### Public Member Functions

- `size_t operator()` (const `::vector< bool, _Alloc > &`) const noexcept

#### 5.798.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 1324 of file `stl_bvector.h`.

The documentation for this struct was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

## 5.799 `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>` Class Template Reference

### Public Types

- `typedef _UIntType result_type`

### Public Member Functions

- [independent\\_bits\\_engine](#) ()
- [independent\\_bits\\_engine](#) (const `_RandomNumberEngine &__rng`)
- [independent\\_bits\\_engine](#) (`_RandomNumberEngine &&__rng`)
- [independent\\_bits\\_engine](#) ([result\\_type \\_\\_s](#))
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_↵`  
same<\_Sseq, `_RandomNumberEngine`>::value> ::type>  
[independent\\_bits\\_engine](#) (`_Sseq &__q`)
- const `_RandomNumberEngine & base` () const noexcept
- void [discard](#) (unsigned long long `__z`)
- [result\\_type operator\(\)](#) ()
- void [seed](#) ()
- void [seed](#) ([result\\_type \\_\\_s](#))
- `template<typename _Sseq >`  
void [seed](#) (`_Sseq &__q`)

### Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

### Friends

- bool [operator==](#) (const [independent\\_bits\\_engine](#) &\_\_lhs, const [independent\\_bits\\_engine](#) &\_\_rhs)
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is,  
[std::independent\\_bits\\_engine](#)< \_RandomNumberEngine, \_\_w, \_UIntType > &\_\_x)

### 5.799.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>  
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.

Definition at line 1059 of file random.h.

### 5.799.2 Member Typedef Documentation

#### 5.799.2.1 result\_type

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>  
typedef _UIntType std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::result\_type
```

The type of the generated random value.

Definition at line 1062 of file random.h.

### 5.799.3 Constructor & Destructor Documentation

#### 5.799.3.1 independent\_bits\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>  
std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::independent\_bits\_engine ( )  
[inline]
```

Constructs a default [independent\\_bits\\_engine](#) engine.

The underlying engine is default constructed as well.

Definition at line 1075 of file random.h.

### 5.799.3.2 `independent_bits_engine()` [2/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    const _RandomNumberEngine & __rng ) [inline], [explicit]
```

Copy constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

#### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1085 of file `random.h`.

### 5.799.3.3 `independent_bits_engine()` [3/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    _RandomNumberEngine && __rng ) [inline], [explicit]
```

Move constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

#### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1095 of file `random.h`.

### 5.799.3.4 `independent_bits_engine()` [4/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    result_type __s ) [inline], [explicit]
```

Seed constructs a `independent_bits_engine` engine.

Constructs the underlying generator engine seeded with `__s`.



**Parameters**

<code>_↵</code>	A seed value for the base class engine.
<code>_s</code>	

Definition at line 1105 of file random.h.

**5.799.3.5 independent\_bits\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, independent_↵
bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    _Sseq & __q ) [inline], [explicit]
```

Generator construct a independent\_bits\_engine engine.

**Parameters**

<code>_↵</code>	A seed sequence.
<code>_q</code>	

Definition at line 1118 of file random.h.

**5.799.4 Member Function Documentation****5.799.4.1 base()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
const _RandomNumberEngine& std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >↵
::base ( ) const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1153 of file random.h.

**5.799.4.2 discard()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard (
    unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Definition at line 1174 of file random.h.

#### 5.799.4.3 max()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max ( ) [inline], [static]
```

Gets the maximum value in the generated random number range.

Definition at line 1167 of file random.h.

#### 5.799.4.4 min()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min ( ) [inline], [static]
```

Gets the minimum value in the generated random number range.

Definition at line 1160 of file random.h.

#### 5.799.4.5 operator()()

```
template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ( )
```

Gets the next value in the generated random number sequence.

Definition at line 742 of file bits/random.tcc.

References std::\_\_lg(), std::numeric\_limits<\_Tp >::max(), and std::numeric\_limits<\_Tp >::min().

#### 5.799.4.6 seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::seed ( ) [inline]
```

Reseeds the independent\_bits\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1127 of file random.h.

**5.799.4.7 seed()** [2/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
    result_type __s ) [inline]
```

Reseeds the independent\_bits\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1135 of file random.h.

**5.799.4.8 seed()** [3/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq >
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
    _Sseq & __q ) [inline]
```

Reseeds the independent\_bits\_engine object with the given seed sequence.

**Parameters**

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 1145 of file random.h.

**5.799.5 Friends And Related Function Documentation****5.799.5.1 operator==**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
bool operator== (
    const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
    const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs ) [friend]
```

Compares two independent\_bits\_engine random number generator objects of the same type for equality.

**Parameters**

<code>__lhs</code>	A independent_bits_engine random number generator object.
<code>__rhs</code>	Another independent_bits_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1199 of file random.h.

**5.799.5.2 `operator>>`**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream<_CharT, _Traits > & __is,
    std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType > & __x ) [friend]
```

Extracts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>independent_bits_engine</code> random number generator engine.

**Returns**

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1217 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.800 `std::indirect_array<_Tp>` Class Template Reference****Public Types**

- `typedef _Tp value_type`

## Public Member Functions

- `indirect_array` (const `indirect_array` &)
- void `operator%=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator%=(const _Expr<_Dom, _Tp> &)` const
- void `operator&=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator&=(const _Expr<_Dom, _Tp> &)` const
- void `operator*=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator*=(const _Expr<_Dom, _Tp> &)` const
- void `operator+=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator+=(const _Expr<_Dom, _Tp> &)` const
- void `operator-=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator-=(const _Expr<_Dom, _Tp> &)` const
- void `operator/=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator/=(const _Expr<_Dom, _Tp> &)` const
- void `operator<=<=` (const `valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator<=<=` (const `_Expr<_Dom, _Tp> &)` const
- `indirect_array` & `operator=` (const `indirect_array` &)
- void `operator=` (const `valarray<_Tp> &)` const
- void `operator=` (const `_Tp` &) const
- template<class \_Dom >  
void `operator=` (const `_Expr<_Dom, _Tp> &)` const
- void `operator>>=` (const `valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator>>=` (const `_Expr<_Dom, _Tp> &)` const
- void `operator^=` (const `valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator^=` (const `_Expr<_Dom, _Tp> &)` const
- void `operator|=` (const `valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator|=` (const `_Expr<_Dom, _Tp> &)` const

## Friends

- class `gslice_array<_Tp>`
- class `valarray<_Tp>`

## 5.800.1 Detailed Description

```
template<class _Tp>
class std::indirect_array<_Tp>
```

Reference to arbitrary subset of an array.

An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

## Parameters

<i>Tp</i>	Element type.
-----------	---------------

Definition at line 84 of file `valarray`.

The documentation for this class was generated from the following files:

- [valarray](#)
- [indirect\\_array.h](#)

5.801 `std::initializer_list<_E>` Class Template Reference

## Public Types

- `typedef const _E * const_iterator`
- `typedef const _E & const_reference`
- `typedef const _E * iterator`
- `typedef const _E & reference`
- `typedef size_t size_type`
- `typedef _E value_type`

## Public Member Functions

- `constexpr const_iterator begin () const noexcept`
- `constexpr const_iterator end () const noexcept`
- `constexpr size_type size () const noexcept`

### 5.801.1 Detailed Description

```
template<class _E>  
class std::initializer_list< _E >
```

initializer\_list

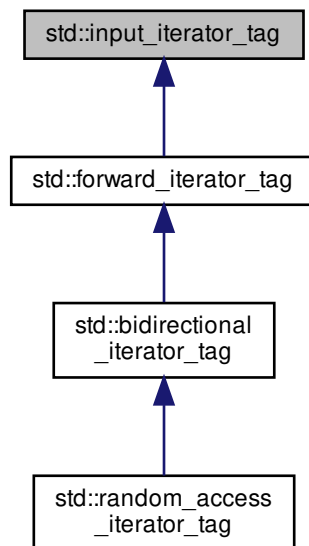
Definition at line 47 of file initializer\_list.

The documentation for this class was generated from the following file:

- [initializer\\_list](#)

### 5.802 std::input\_iterator\_tag Struct Reference

Inheritance diagram for std::input\_iterator\_tag:



### 5.802.1 Detailed Description

Marking input iterators.

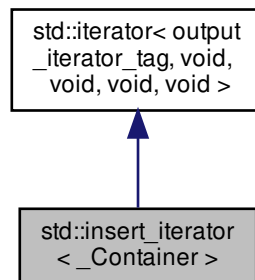
Definition at line 89 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.803 std::insert\_iterator&lt; \_Container &gt; Class Template Reference

Inheritance diagram for std::insert\_iterator< \_Container >:



## Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

## Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator` & `operator*` ()
- `insert_iterator` & `operator++` ()
- `insert_iterator` & `operator++` (int)
- `insert_iterator` & `operator=` (const `typename _Container::value_type &__value`)
- `insert_iterator` & `operator=` (`typename _Container::value_type &&__value`)

## Protected Attributes

- `_Container *` **`container`**
- `_Container::iterator` **`iter`**



### 5.803.1 Detailed Description

```
template<typename _Container>
class std::insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 642 of file bits/stl\_iterator.h.

### 5.803.2 Member Typedef Documentation

#### 5.803.2.1 container\_type

```
template<typename _Container >
typedef _Container std::insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

Definition at line 651 of file bits/stl\_iterator.h.

#### 5.803.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

#### 5.803.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >←
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

#### 5.803.2.4 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

#### 5.803.2.5 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

#### 5.803.2.6 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

### 5.803.3 Constructor & Destructor Documentation

#### 5.803.3.1 insert\_iterator()

```
template<typename _Container >
std::insert_iterator< _Container >::insert_iterator (
    _Container & __x,
    typename _Container::iterator __i ) [inline]
```

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 657 of file bits/stl\_iterator.h.

#### 5.803.4 Member Function Documentation

#### 5.803.4.1 operator\*()

```
template<typename _Container >
insert_iterator& std::insert_iterator< _Container >::operator* ( ) [inline]
```

Simply returns \*this.

Definition at line 711 of file bits/stl\_iterator.h.

#### 5.803.4.2 operator++() [1/2]

```
template<typename _Container >
insert_iterator& std::insert_iterator< _Container >::operator++ ( ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 716 of file bits/stl\_iterator.h.

#### 5.803.4.3 operator++() [2/2]

```
template<typename _Container >
insert_iterator& std::insert_iterator< _Container >::operator++ (
    int ) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 721 of file bits/stl\_iterator.h.

#### 5.803.4.4 operator=()

```
template<typename _Container >
insert_iterator& std::insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value ) [inline]
```

##### Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> .
----------------------	---

##### Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;
// vector v contains A, 1, 2, 3, and Z
```

Definition at line 693 of file bits/stl\_iterator.h.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 5.804 std::integer\_sequence< \_Tp, \_Idx > Struct Template Reference

### Public Types

- typedef \_Tp **value\_type**

### Static Public Member Functions

- static constexpr size\_t **size** () noexcept

#### 5.804.1 Detailed Description

```
template<typename _Tp, _Tp... _Idx>
struct std::integer_sequence< _Tp, _Idx >
```

Class template integer\_sequence.

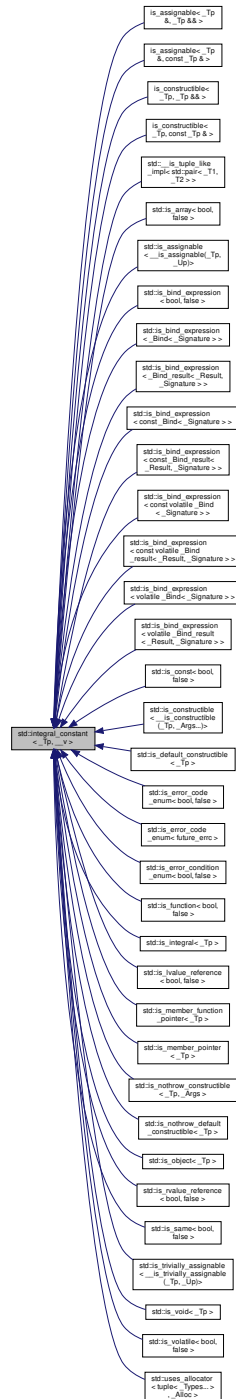
Definition at line 317 of file utility.

The documentation for this struct was generated from the following file:

- [utility](#)

## 5.805 std::integral\_constant< \_Tp, \_\_v > Struct Template Reference

Inheritance diagram for std::integral\_constant< \_Tp, \_\_v >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.805.1 Detailed Description

```
template<typename _Tp, _Tp __v>  
struct std::integral_constant< _Tp, __v >
```

[integral\\_constant](#)

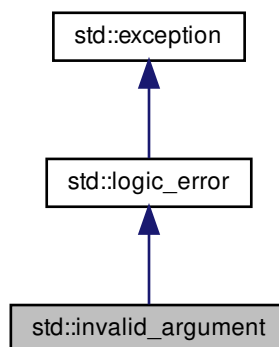
Definition at line 57 of file [type\\_traits](#).

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.806 std::invalid\_argument Class Reference

Inheritance diagram for std::invalid\_argument:



- ### 5.806.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 158 of file stdexcept.

## 5.806.2 Member Function Documentation

### 5.806.2.1 what()

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

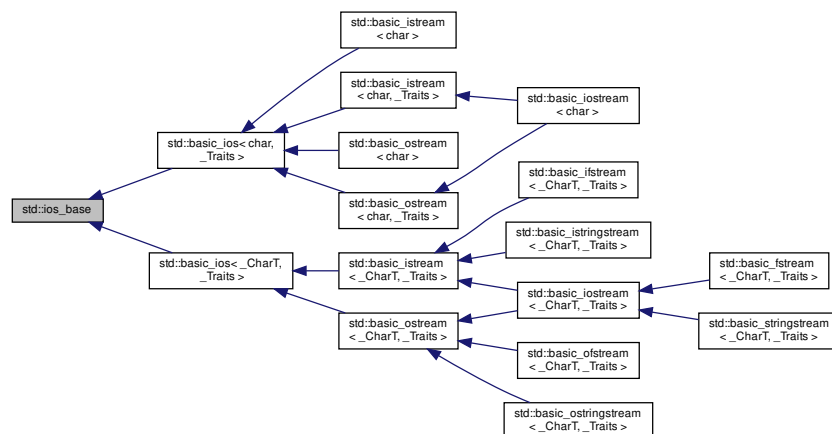
Reimplemented in `std::future_error`.

The documentation for this class was generated from the following file:

- `stdexcept`

## 5.807 `std::ios_base` Class Reference

Inheritance diagram for `std::ios_base`:



## Classes

- class [failure](#)

## Public Types

- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef [\\_ios\\_Fmtflags](#) [fmtflags](#)
- typedef int [io\\_state](#)
- typedef [\\_ios\\_istate](#) [iostate](#)
- typedef int [open\\_mode](#)
- typedef [\\_ios\\_Openmode](#) [openmode](#)
- typedef int [seek\\_dir](#)
- typedef [\\_ios\\_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)

## Public Member Functions

- [ios\\_base](#) (const [ios\\_base](#) &)=delete
- virtual [~ios\\_base](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc) throw ()
- long & [iword](#) (int \_\_ix)
- [ios\\_base](#) & [operator=](#) (const [ios\\_base](#) &)=delete
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)

## Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()



### Static Public Attributes

- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iosstate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iosstate](#) eofbit
- static const [iosstate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iosstate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { **`_S_local_word_size`** }

### Protected Member Functions

- void **`_M_call_callbacks`** ([event](#) \_\_ev) throw ()
- void **`_M_dispose_callbacks`** (void) throw ()
- [\\_Words](#) & **`_M_grow_words`** (int \_\_index, bool \_\_iword)
- void **`_M_init`** () throw ()
- void **`_M_move`** ([ios\\_base](#) &) noexcept
- void **`_M_swap`** ([ios\\_base](#) &\_\_rhs) noexcept

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `iostate _M_exception`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [ _S_local_word_size]`
- `streamsize _M_precision`
- `iostate _M_streambuf_state`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 5.807.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 228 of file `ios_base.h`.

## 5.807.2 Member Typedef Documentation

## 5.807.2.1 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

## Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 506 of file `ios_base.h`.

### 5.807.2.2 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 323 of file `ios_base.h`.

### 5.807.2.3 iostate

```
typedef _Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 398 of file `ios_base.h`.

#### 5.807.2.4 openmode

```
typedef _Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 429 of file `ios_base.h`.

#### 5.807.2.5 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 461 of file `ios_base.h`.

### 5.807.3 Member Enumeration Documentation

#### 5.807.3.1 event

```
enum std::ios_base::event
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 489 of file `ios_base.h`.

## 5.807.4 Constructor & Destructor Documentation

### 5.807.4.1 ~ios\_base()

```
virtual std::ios_base::~~ios_base ( ) [virtual]
```

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

## 5.807.5 Member Function Documentation

### 5.807.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc ( ) const [inline]
```

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 776 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

### 5.807.5.2 flags() [1/2]

```
fmtflags std::ios_base::flags ( ) const [inline]
```

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 621 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

### 5.807.5.3 flags() [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl ) [inline]
```

Setting new format flags all at once.

## Parameters

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

## Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 632 of file `ios_base.h`.

## 5.807.5.4 getloc()

```
locale std::ios_base::getloc ( ) const [inline]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 765 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, and `std::ws()`.

## 5.807.5.5 imbue()

```
locale std::ios_base::imbue (
    const locale & __loc ) throw ( )
```

Setting a new locale.

## Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

## Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with `imbue_event`.

Referenced by `std::basic_ios< char, _Traits >::imbue()`.

#### 5.807.5.6 `iword()`

```
long& std::ios_base::iword (
    int __ix ) [inline]
```

Access to integer array.

##### Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 811 of file `ios_base.h`.

#### 5.807.5.7 `precision()` [1/2]

```
streamsize std::ios_base::precision ( ) const [inline]
```

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 691 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 5.807.5.8 `precision()` [2/2]

```
streamsize std::ios_base::precision (
    streamsize __prec ) [inline]
```

Changing flags.

## Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

## Returns

The previous value of `precision()`.

Definition at line 700 of file `ios_base.h`.

## 5.807.5.9 pword()

```
void*& std::ios_base::pword (
    int __ix ) [inline]
```

Access to void pointer array.

## Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

## Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 832 of file `ios_base.h`.

## 5.807.5.10 register\_callback()

```
void std::ios_base::register_callback (
    event_callback __fn,
    int __index )
```

Add the callback `__fn` with parameter `__index`.



**Parameters**

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.807.5.11** `setf()` [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl ) [inline]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 648 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**5.807.5.12** `setf()` [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask ) [inline]
```

Setting new format flags.

**Parameters**

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 665 of file `ios_base.h`.

#### 5.807.5.13 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
    bool __sync = true ) [static]
```

Interaction with the standard C I/O objects.

##### Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 5.807.5.14 unsetf()

```
void std::ios_base::unsetf (
    fmtflags __mask ) [inline]
```

Clearing format flags.

##### Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 680 of file `ios_base.h`.

#### 5.807.5.15 width() [1/2]

```
streamsize std::ios_base::width ( ) const [inline]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 714 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**5.807.5.16 width()** [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide ) [inline]
```

Changing flags.

**Parameters**

<code>__wide</code>	The new width value.
---------------------	----------------------

**Returns**

The previous value of `width()`.

Definition at line 723 of file `ios_base.h`.

**5.807.5.17 xalloc()**

```
static int std::ios_base::xalloc ( ) throw ( ) [static]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.807.6 Member Data Documentation

### 5.807.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 378 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

### 5.807.6.2 app

```
const openmode std::ios_base::app [static]
```

Seek to end before each write.

Definition at line 432 of file `ios_base.h`.

### 5.807.6.3 ate

```
const openmode std::ios_base::ate [static]
```

Open and seek to end immediately after opening.

Definition at line 435 of file `ios_base.h`.

### 5.807.6.4 badbit

```
const iostate std::ios_base::badbit [static]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 402 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 5.807.6.5 basefield

```
const fmtflags std::ios_base::basefield [static]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 381 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.807.6.6 beg

```
const seekdir std::ios_base::beg [static]
```

Request a seek relative to the beginning of the stream.

Definition at line 464 of file `ios_base.h`.

#### 5.807.6.7 binary

```
const openmode std::ios_base::binary [static]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 440 of file `ios_base.h`.

#### 5.807.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 326 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 5.807.6.9 cur

```
const seekdir std::ios_base::cur [static]
```

Request a seek relative to the current position within the sequence.

Definition at line 467 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 5.807.6.10 dec

```
const fmtflags std::ios_base::dec [static]
```

Converts integer input or generates integer output in decimal base.

Definition at line 329 of file ios\_base.h.

#### 5.807.6.11 end

```
const seekdir std::ios_base::end [static]
```

Request a seek relative to the current end of the sequence.

Definition at line 470 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 5.807.6.12 eofbit

```
const iostate std::ios_base::eofbit [static]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 405 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 5.807.6.13 failbit

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 410 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry<::sentry()`.

#### 5.807.6.14 fixed

```
const fmtflags std::ios_base::fixed [static]
```

Generate floating-point output in fixed-point notation.

Definition at line 332 of file ios\_base.h.

#### 5.807.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 384 of file ios\_base.h.

#### 5.807.6.16 goodbit

```
const iostate std::ios_base::goodbit [static]
```

Indicates all is well.

Definition at line 413 of file ios\_base.h.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

#### 5.807.6.17 hex

```
const fmtflags std::ios_base::hex [static]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 335 of file ios\_base.h.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< char >::operator<<()`.

#### 5.807.6.18 in

```
const openmode std::ios_base::in [static]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 443 of file ios\_base.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

#### 5.807.6.19 internal

```
const fmtflags std::ios_base::internal [static]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 340 of file ios\_base.h.

#### 5.807.6.20 left

```
const fmtflags std::ios_base::left [static]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 344 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.



**5.807.6.21 oct**

```
const fmtflags std::ios_base::oct [static]
```

Converts integer input or generates integer output in octal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::operator<<().

**5.807.6.22 out**

```
const openmode std::ios_base::out [static]
```

Open for output. Default for ofstream and fstream.

Definition at line 446 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos().

**5.807.6.23 right**

```
const fmtflags std::ios_base::right [static]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 351 of file ios\_base.h.

**5.807.6.24 scientific**

```
const fmtflags std::ios_base::scientific [static]
```

Generates floating-point output in scientific notation.

Definition at line 354 of file ios\_base.h.

**5.807.6.25 showbase**

```
const fmtflags std::ios_base::showbase [static]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 358 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put().

#### 5.807.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 362 of file ios\_base.h.

#### 5.807.6.27 showpos

```
const fmtflags std::ios_base::showpos [static]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 365 of file ios\_base.h.

#### 5.807.6.28 skipws

```
const fmtflags std::ios_base::skipws [static]
```

Skips leading white space before certain input operations.

Definition at line 368 of file ios\_base.h.

#### 5.807.6.29 trunc

```
const openmode std::ios_base::trunc [static]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 449 of file ios\_base.h.

#### 5.807.6.30 unitbuf

```
const fmtflags std::ios_base::unitbuf [static]
```

Flushes output after each output operation.

Definition at line 371 of file ios\_base.h.

### 5.807.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 375 of file `ios_base.h`.

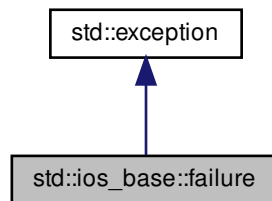
Referenced by `std::num_put<_CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 5.808 std::ios\_base::failure Class Reference

Inheritance diagram for `std::ios_base::failure`:



### Public Member Functions

- **failure** (const [string](#) &\_\_str) throw ()
- virtual const char \* [what](#) () const throw ()

### 5.808.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

Definition at line 276 of file `ios_base.h`.

## 5.808.2 Member Function Documentation

## 5.808.2.1 what()

```
virtual const char* std::ios_base::failure::what ( ) const throw ( ) [virtual]
```

Returns a C-style character string describing the general cause of the current error.

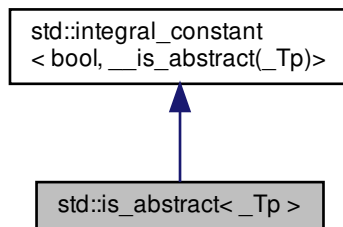
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 5.809 std::is\_abstract&lt;\_Tp&gt; Struct Template Reference

Inheritance diagram for std::is\_abstract<\_Tp>:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ( ) const noexcept
- constexpr value\_type **operator()** ( ) const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 5.809.1 Detailed Description

```
template<typename _Tp>
struct std::is_abstract< _Tp >
```

is\_abstract

Definition at line 713 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.810 std::is\_arithmetic< \_Tp > Struct Template Reference

Inherits type< is\_integral< \_Tp >, is\_floating\_point< \_Tp > >.

#### 5.810.1 Detailed Description

```
template<typename _Tp>
struct std::is_arithmetic< _Tp >
```

is\_arithmetic

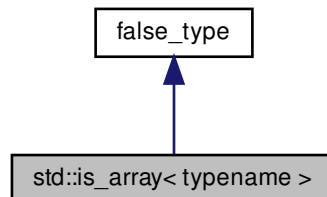
Definition at line 575 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.811 std::is\_array< typename > Struct Template Reference

Inheritance diagram for std::is\_array< typename >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.811.1 Detailed Description

```
template<typename>
struct std::is_array< typename >
```

is\_array

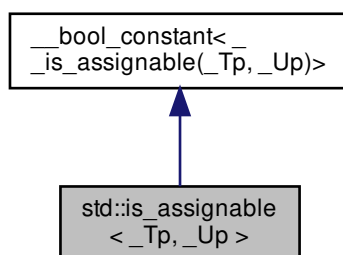
Definition at line 347 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.812 std::is\_assignable&lt; \_Tp, \_Up &gt; Struct Template Reference

Inheritance diagram for std::is\_assignable< \_Tp, \_Up >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.812.1 Detailed Description

```
template<typename _Tp, typename _Up>  
struct std::is_assignable< _Tp, _Up >
```

is\_assignable

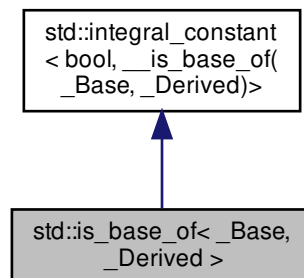
Definition at line 999 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.813 std::is\_base\_of< \_Base, \_Derived > Struct Template Reference

Inheritance diagram for std::is\_base\_of< \_Base, \_Derived >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 5.813.1 Detailed Description

```
template<typename _Base, typename _Derived>  
struct std::is_base_of< _Base, _Derived >
```

is\_base\_of

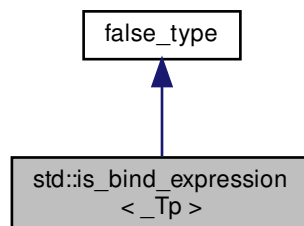
Definition at line 1288 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.814 std::is\_bind\_expression&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_bind\_expression< \_Tp >:





### Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.814.1 Detailed Description

```
template<typename _Tp>
struct std::is_bind_expression<_Tp >
```

Determines if the given type \_Tp is a function object that should be treated as a subexpression when evaluating calls to function objects returned by bind().

C++11 [func.bind.isbind].

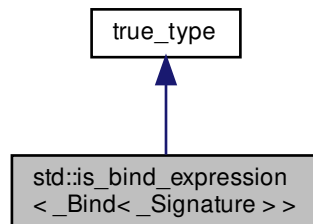
Definition at line 174 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 5.815 std::is\_bind\_expression<\_Bind<\_Signature > > Struct Template Reference

Inheritance diagram for std::is\_bind\_expression<\_Bind<\_Signature > >:



## Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v> **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.815.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression<_Bind<_Signature>>>
```

Class template `_Bind` is always a bind expression.

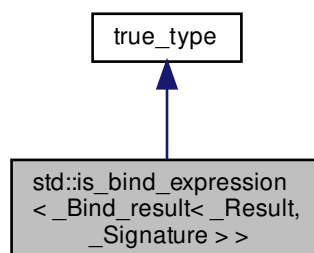
Definition at line 693 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.816 `std::is_bind_expression<_Bind_result<_Result, _Signature>>` Struct Template Reference

Inheritance diagram for `std::is_bind_expression<_Bind_result<_Result, _Signature>>`:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.816.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

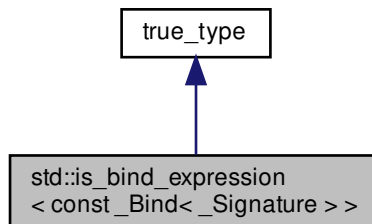
Definition at line 725 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 5.817 `std::is_bind_expression< const _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const _Bind< _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.817.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const _Bind< _Signature > >
```

Class template \_Bind is always a bind expression.

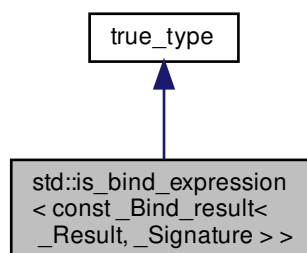
Definition at line 701 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.818 std::is\_bind\_expression&lt; const \_Bind\_result&lt; \_Result, \_Signature &gt; &gt; Struct Template Reference

Inheritance diagram for std::is\_bind\_expression< const \_Bind\_result< \_Result, \_Signature > >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.818.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

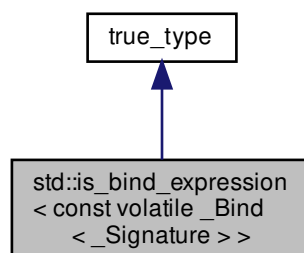
Definition at line 733 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 5.819 `std::is_bind_expression< const volatile _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const volatile _Bind< _Signature > >`:



## Public Types

- typedef `integral_constant< _Tp, __v >` **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr `_Tp` **value**

## 5.819.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

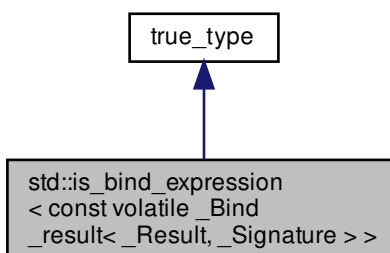
Definition at line 717 of file `functional`.

The documentation for this struct was generated from the following file:

- `functional`

5.820 `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.820.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

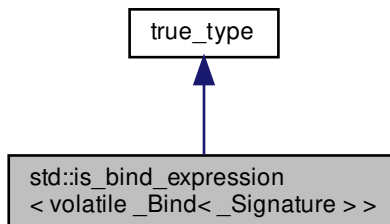
Definition at line 749 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 5.821 `std::is_bind_expression< volatile _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< volatile _Bind< _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.821.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< volatile _Bind< _Signature > >
```

Class template \_Bind is always a bind expression.

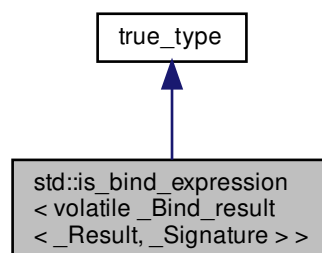
Definition at line 709 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.822 std::is\_bind\_expression&lt; volatile \_Bind\_result&lt; \_Result, \_Signature &gt; &gt; Struct Template Reference

Inheritance diagram for std::is\_bind\_expression< volatile \_Bind\_result< \_Result, \_Signature > >:





#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.822.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

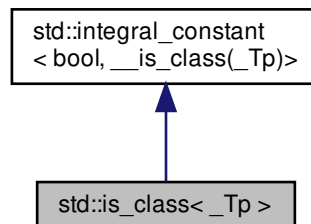
Definition at line 741 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 5.823 `std::is_class< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_class< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 5.823.1 Detailed Description

```
template<typename _Tp>  
struct std::is_class<_Tp>
```

is\_class

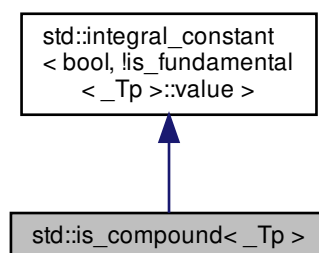
Definition at line 437 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.824 std::is\_compound&lt;\_Tp&gt; Struct Template Reference

Inheritance diagram for std::is\_compound<\_Tp>:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr bool **value**

#### 5.824.1 Detailed Description

```
template<typename _Tp>  
struct std::is_compound< _Tp >
```

is\_compound

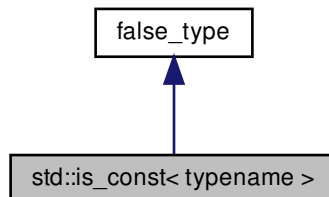
Definition at line 605 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.825 std::is\_const< typename > Struct Template Reference

Inheritance diagram for std::is\_const< typename >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.825.1 Detailed Description

```
template<typename>
struct std::is_const< typename >
```

is\_const

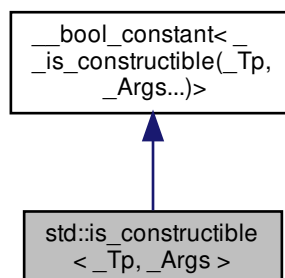
Definition at line 643 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.826 std::is\_constructible&lt; \_Tp, \_Args &gt; Struct Template Reference

Inheritance diagram for std::is\_constructible< \_Tp, \_Args >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.826.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_constructible< _Tp, _Args >
```

is\_constructible

Definition at line 872 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.827 std::is\_convertible< \_From, \_To > Struct Template Reference

Inherits type< \_From, \_To >.

#### 5.827.1 Detailed Description

```
template<typename _From, typename _To>
struct std::is_convertible< _From, _To >
```

is\_convertible

Definition at line 1320 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.828 `std::is_copy_assignable<_Tp>` Struct Template Reference**

Inherits `std::__is_copy_assignable_impl<_Tp, bool>`.

**5.828.1 Detailed Description**

```
template<typename _Tp>
struct std::is_copy_assignable<_Tp>
```

`is_copy_assignable`

Definition at line 1017 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.829 `std::is_copy_constructible<_Tp>` Struct Template Reference**

Inherits `std::__is_copy_constructible_impl<_Tp, bool>`.

Inherited by `std::__is_copy_insertable< allocator<_Tp>>`.

**5.829.1 Detailed Description**

```
template<typename _Tp>
struct std::is_copy_constructible<_Tp>
```

`is_copy_constructible`

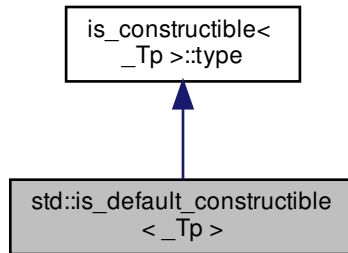
Definition at line 896 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.830 `std::is_default_constructible< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_default_constructible< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.830.1 Detailed Description

```
template<typename _Tp>
struct std::is_default_constructible< _Tp >
```

`is_default_constructible`

Definition at line 878 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.831 `std::is_destructible< _Tp >` Struct Template Reference

Inherits `type< _Tp >`.

## 5.831.1 Detailed Description

```
template<typename _Tp>
struct std::is_destructible< _Tp >
```

`is_destructible`

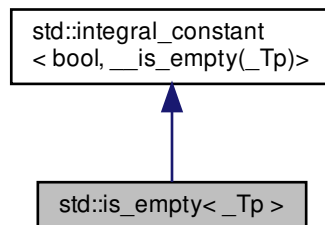
Definition at line 818 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.832 `std::is_empty< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_empty< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept



#### Static Public Attributes

- static constexpr bool **value**

#### 5.832.1 Detailed Description

```
template<typename _Tp>  
struct std::is_empty< _Tp >
```

is\_empty

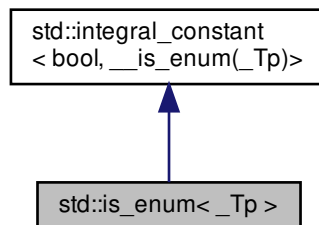
Definition at line 692 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.833 std::is\_enum< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_enum< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 5.833.1 Detailed Description

```
template<typename _Tp>  
struct std::is_enum< _Tp >
```

is\_enum

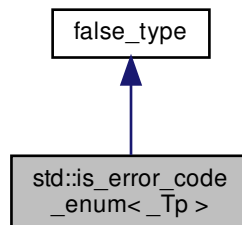
Definition at line 425 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.834 std::is\_error\_code\_enum< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_error\_code\_enum< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.834.1 Detailed Description

```
template<typename _Tp>  
struct std::is_error_code_enum< _Tp >
```

`is_error_code_enum`

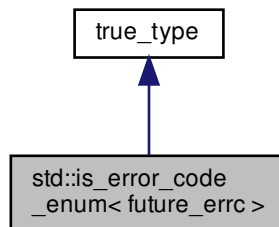
Definition at line 53 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 5.835 `std::is_error_code_enum< future_errc >` Struct Template Reference

Inheritance diagram for `std::is_error_code_enum< future_errc >`:



#### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.835.1 Detailed Description

```
template<>
struct std::is_error_code_enum< future_errc >
```

Specialization.

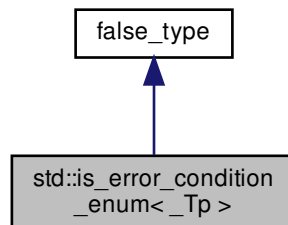
Definition at line 76 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

#### 5.836 std::is\_error\_condition\_enum< \_Tp > Struct Template Reference

Inheritance diagram for `std::is_error_condition_enum< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.836.1 Detailed Description

```
template<typename _Tp>  
struct std::is_error_condition_enum< _Tp >
```

`is_error_condition_enum`

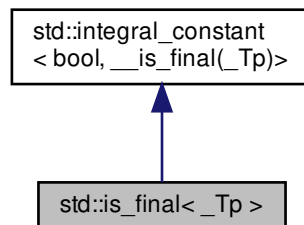
Definition at line 57 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 5.837 `std::is_final< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_final< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< bool, `__v` > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 5.837.1 Detailed Description

```
template<typename _Tp>
struct std::is_final< _Tp >
```

is\_final

Definition at line 706 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.838 std::is\_floating\_point&lt; \_Tp &gt; Struct Template Reference

Inherits type< remove\_cv< \_Tp >::type >.

## 5.838.1 Detailed Description

```
template<typename _Tp>
struct std::is_floating_point< _Tp >
```

is\_floating\_point

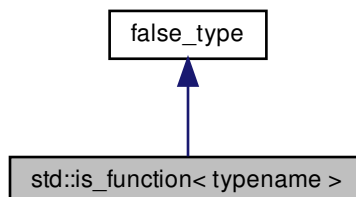
Definition at line 341 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.839 std::is\_function&lt; typename &gt; Struct Template Reference

Inheritance diagram for std::is\_function< typename >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.839.1 Detailed Description

```
template<typename>  
struct std::is_function< typename >
```

is\_function

Definition at line 391 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.840 std::is\_fundamental< \_Tp > Struct Template Reference

Inherits type< is\_arithmetic< \_Tp >, is\_void< \_Tp >, is\_null\_pointer< \_Tp > >.

#### 5.840.1 Detailed Description

```
template<typename _Tp>  
struct std::is_fundamental< _Tp >
```

is\_fundamental

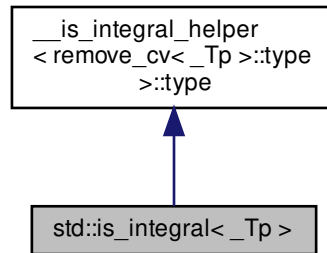
Definition at line 581 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.841 `std::is_integral<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_integral<_Tp>`:



## Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.841.1 Detailed Description

```
template<typename _Tp>  
struct std::is_integral<_Tp>
```

`is_integral`

Definition at line 313 of file `type_traits`.

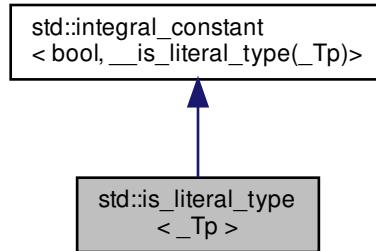
The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 5.842 std::is\_literal\_type< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_literal\_type< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr bool **value**

### 5.842.1 Detailed Description

```
template<typename _Tp>
struct std::is_literal_type< _Tp >
```

is\_literal\_type

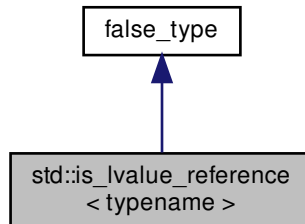
Definition at line 686 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.843 `std::is_lvalue_reference< typename >` Struct Template Reference

Inheritance diagram for `std::is_lvalue_reference< typename >`:

**Public Types**

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

**Static Public Attributes**

- static constexpr `_Tp` **value**

## 5.843.1 Detailed Description

```
template<typename>  
struct std::is_lvalue_reference< typename >
```

`is_lvalue_reference`

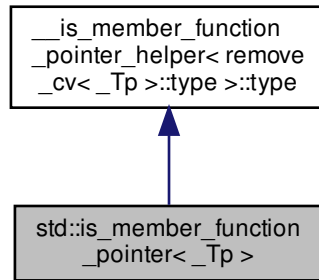
Definition at line 374 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.844 `std::is_member_function_pointer< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_member_function_pointer< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.844.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_function_pointer< _Tp >
```

`is_member_function_pointer`

Definition at line 418 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.845 std::is\_member\_object\_pointer&lt; \_Tp &gt; Struct Template Reference

Inherits type< remove\_cv< \_Tp >::type >.

## 5.845.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_object_pointer< _Tp >
```

is\_member\_object\_pointer

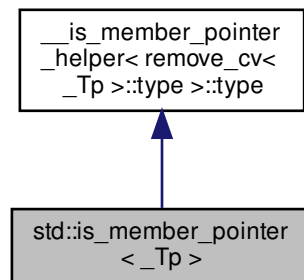
Definition at line 403 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.846 std::is\_member\_pointer&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_member\_pointer< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.846.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_pointer< _Tp >
```

is\_member\_pointer

Definition at line 594 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.847 std::is\_move\_assignable< \_Tp > Struct Template Reference

Inherits std::\_\_is\_move\_assignable\_impl< \_Tp, bool >.

#### 5.847.1 Detailed Description

```
template<typename _Tp>
struct std::is_move_assignable< _Tp >
```

is\_move\_assignable

Definition at line 1035 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.848 std::is\_move\_constructible< \_Tp > Struct Template Reference

Inherits std::\_\_is\_move\_constructible\_impl< \_Tp, bool >.

## 5.848.1 Detailed Description

```
template<typename _Tp>
struct std::is_move_constructible< _Tp >
```

`is_move_constructible`

Definition at line 914 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.849 `std::is_nothrow_assignable<_Tp, _Up>` Struct Template Reference

Inherits `std::__and_<...>`.

## 5.849.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_nothrow_assignable< _Tp, _Up >
```

`is_nothrow_assignable`

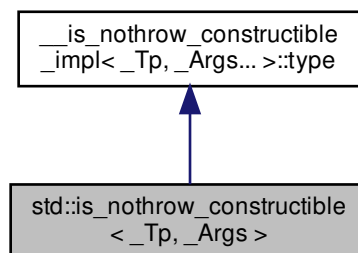
Definition at line 1046 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.850 `std::is_nothrow_constructible<_Tp, _Args>` Struct Template Reference

Inheritance diagram for `std::is_nothrow_constructible<_Tp, _Args>`:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.850.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_nothrow_constructible< _Tp, _Args >
```

is\_nothrow\_constructible

Definition at line 950 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.851 std::is\_nothrow\_copy\_assignable< \_Tp > Struct Template Reference

Inherits std::\_\_is\_nt\_copy\_assignable\_impl< \_Tp, bool >.

#### 5.851.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_assignable< _Tp >
```

is\_nothrow\_copy\_assignable

Definition at line 1065 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.852 std::is\_nothrow\_copy\_constructible&lt; \_Tp &gt; Struct Template Reference

Inherits std::\_\_is\_nothrow\_copy\_constructible\_impl< \_Tp, bool >.

## 5.852.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_constructible< _Tp >
```

is\_nothrow\_copy\_constructible

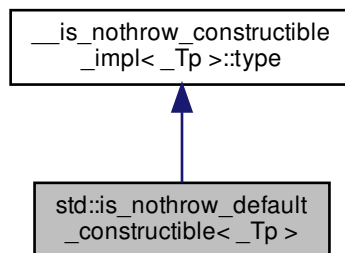
Definition at line 975 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.853 std::is\_nothrow\_default\_constructible&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_nothrow\_default\_constructible< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept



#### Static Public Attributes

- static constexpr `_Tp` **value**

##### 5.853.1 Detailed Description

```
template<typename _Tp>  
struct std::is_nothrow_default_constructible< _Tp >
```

`is_nothrow_default_constructible`

Definition at line 956 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.854 `std::is_nothrow_destructible< _Tp >` Struct Template Reference

Inherits `type< _Tp >`.

##### 5.854.1 Detailed Description

```
template<typename _Tp>  
struct std::is_nothrow_destructible< _Tp >
```

`is_nothrow_destructible`

Definition at line 866 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.855 `std::is_nothrow_move_assignable< _Tp >` Struct Template Reference

Inherits `std::__is_nt_move_assignable_impl< _Tp, bool >`.

## 5.855.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_assignable<_Tp>
```

`is_nothrow_move_assignable`

Definition at line 1083 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.856 `std::is_nothrow_move_constructible<_Tp>` Struct Template Reference

Inherits `std::__is_nothrow_move_constructible_impl<_Tp, bool>`.

## 5.856.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_constructible<_Tp>
```

`is_nothrow_move_constructible`

Definition at line 993 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.857 `std::is_nothrow_swappable<_Tp>` Struct Template Reference

Inherits `type<_Tp>`.

## 5.857.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_swappable<_Tp>
```

`is_nothrow_swappable`

Definition at line 2465 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.858 `std::is_nothrow_swappable_with<_Tp, _Up>` Struct Template Reference

Inherits `type<_Tp, _Up>`.

#### 5.858.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_nothrow_swappable_with<_Tp, _Up>
```

`is_nothrow_swappable_with`

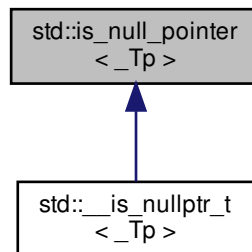
Definition at line 2549 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.859 `std::is_null_pointer<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_null_pointer<_Tp>`:



#### 5.859.1 Detailed Description

```
template<typename _Tp>
struct std::is_null_pointer<_Tp>
```

`is_null_pointer` (LWG 2247).

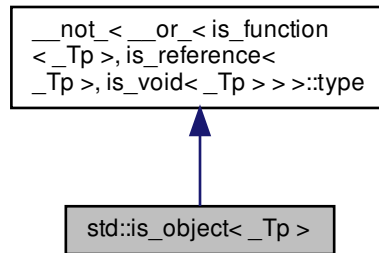
Definition at line 554 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.860 std::is\_object&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_object< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 5.860.1 Detailed Description

```
template<typename _Tp>
struct std::is_object< _Tp >
```

is\_object

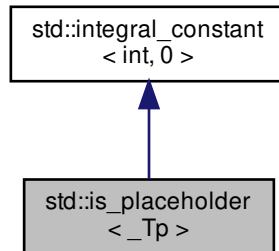
Definition at line 588 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.861 `std::is_placeholder< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_placeholder< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< int, \_\_v > **type**
- typedef int **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr int **value**

### 5.861.1 Detailed Description

```
template<typename _Tp>
struct std::is_placeholder< _Tp >
```

Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is.

C++11 [func.bind.isplace].

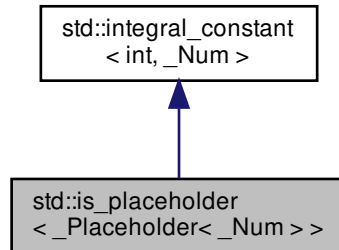
Definition at line 185 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.862 std::is\_placeholder&lt; \_Placeholder&lt; \_Num &gt; &gt; Struct Template Reference

Inheritance diagram for std::is\_placeholder< \_Placeholder< \_Num > >:



## Public Types

- typedef [integral\\_constant](#)< int, \_\_v > **type**
- typedef int **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr int **value**

## 5.862.1 Detailed Description

```
template<int _Num>
struct std::is_placeholder< _Placeholder< _Num > >
```

Partial specialization of is\_placeholder that provides the placeholder number for the placeholder objects defined by libstdc++.

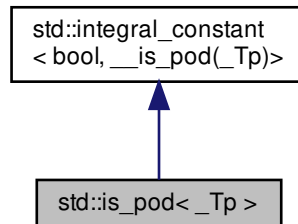
Definition at line 248 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.863 `std::is_pod<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_pod<_Tp>`:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 5.863.1 Detailed Description

```
template<typename _Tp>  
struct std::is_pod<_Tp>
```

`is_pod`

Definition at line 680 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.864 std::is\_pointer&lt; \_Tp &gt; Struct Template Reference

Inherits type< remove\_cv< \_Tp >::type >.

## 5.864.1 Detailed Description

```
template<typename _Tp>
struct std::is_pointer< _Tp >
```

is\_pointer

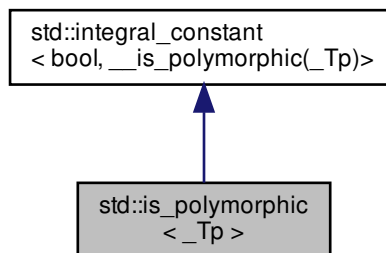
Definition at line 368 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.865 std::is\_polymorphic&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_polymorphic< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept



#### Static Public Attributes

- static constexpr bool **value**

#### 5.865.1 Detailed Description

```
template<typename _Tp>
struct std::is_polymorphic< _Tp >
```

is\_polymorphic

Definition at line 698 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.866 std::is\_reference< \_Tp > Struct Template Reference

Inherits type< is\_lvalue\_reference< \_Tp >, is\_rvalue\_reference< \_Tp > >.

#### 5.866.1 Detailed Description

```
template<typename _Tp>
struct std::is_reference< _Tp >
```

is\_reference

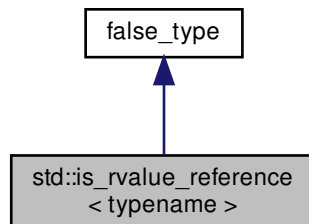
Definition at line 568 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.867 std::is\_rvalue\_reference< typename > Struct Template Reference

Inheritance diagram for std::is\_rvalue\_reference< typename >:



## Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr `_Tp` **value**

## 5.867.1 Detailed Description

```
template<typename>  
struct std::is_rvalue_reference< typename >
```

`is_rvalue_reference`

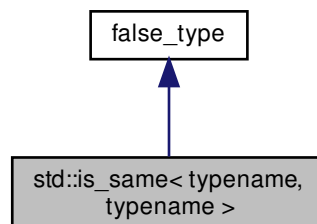
Definition at line 383 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.868 `std::is_same< typename, typename >` Struct Template Reference

Inheritance diagram for `std::is_same< typename, typename >`:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.868.1 Detailed Description

```
template<typename, typename>
struct std::is_same< typename, typename >
```

is\_same

Definition at line 1279 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.869 std::is\_scalar< \_Tp > Struct Template Reference

Inherits type< is\_arithmetic< \_Tp >, is\_enum< \_Tp >, is\_pointer< \_Tp >, is\_member\_pointer< \_Tp >, is\_null\_↵  
pointer< \_Tp > >.

#### 5.869.1 Detailed Description

```
template<typename _Tp>
struct std::is_scalar< _Tp >
```

is\_scalar

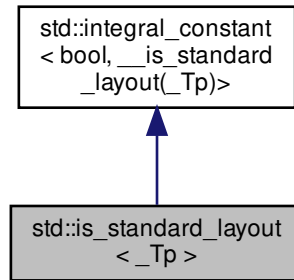
Definition at line 598 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.870 std::is\_standard\_layout&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_standard\_layout< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 5.870.1 Detailed Description

```
template<typename _Tp>
struct std::is_standard_layout< _Tp >
```

is\_standard\_layout

Definition at line 673 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.871 `std::is_swappable< _Tp >` Struct Template Reference

Inherits type< \_Tp >.

### 5.871.1 Detailed Description

```
template<typename _Tp>
struct std::is_swappable< _Tp >
```

Metafunctions used for detecting swappable types: p0185r1.

`is_swappable`

Definition at line 2459 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.872 `std::is_swappable_with< _Tp, _Up >` Struct Template Reference

Inherits type< \_Tp, \_Up >.

### 5.872.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_swappable_with< _Tp, _Up >
```

`is_swappable_with`

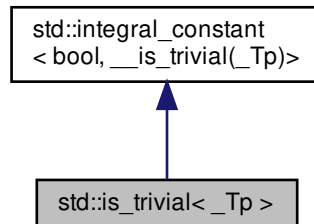
Definition at line 2543 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.873 std::is\_trivial< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_trivial< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr bool **value**

#### 5.873.1 Detailed Description

```
template<typename _Tp>  
struct std::is_trivial< _Tp >
```

is\_trivial

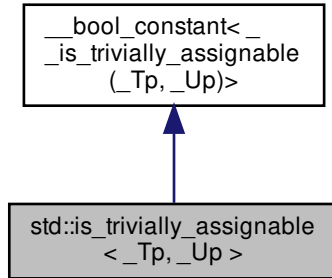
Definition at line 661 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.874 std::is\_trivially\_assignable< \_Tp, \_Up > Struct Template Reference

Inheritance diagram for std::is\_trivially\_assignable< \_Tp, \_Up >:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.874.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_trivially_assignable< _Tp, _Up >
```

is\_trivially\_assignable

Definition at line 1174 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.875 `std::is_trivially_constructible<_Tp, _Args>` Struct Template Reference

Inherits type< `is_constructible<_Tp, _Args...>`, `__bool_constant<__is_trivially_constructible(_Tp, _Args...)>`>.

### 5.875.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_trivially_constructible<_Tp, _Args>
```

`is_trivially_constructible`

Definition at line 1089 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.876 `std::is_trivially_default_constructible<_Tp>` Struct Template Reference

Inherits type< `_Tp`>.

### 5.876.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_default_constructible<_Tp>
```

`is_trivially_default_constructible`

Definition at line 1096 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.877 `std::is_trivially_destructible<_Tp>` Struct Template Reference

Inherits `std::__and<...>`.



### 5.877.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_destructible< _Tp >
```

is\_trivially\_destructible

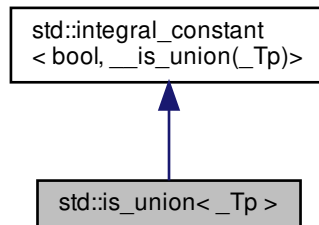
Definition at line 1222 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.878 std::is\_union< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_union< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

## 5.878.1 Detailed Description

```
template<typename _Tp>
struct std::is_union< _Tp >
```

is\_union

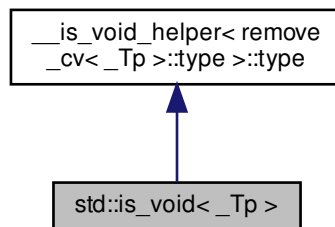
Definition at line 431 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.879 std::is\_void&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_void< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

### 5.879.1 Detailed Description

```
template<typename _Tp>
struct std::is_void< _Tp >
```

is\_void

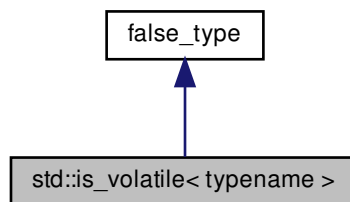
Definition at line 202 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.880 std::is\_volatile< typename > Struct Template Reference

Inheritance diagram for std::is\_volatile< typename >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

## 5.880.1 Detailed Description

```
template<typename>
struct std::is_volatile< typename >
```

is\_volatile

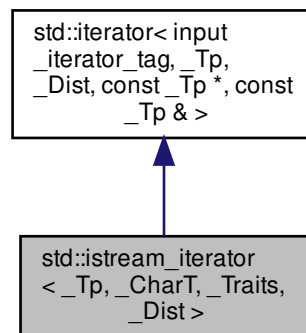
Definition at line 652 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.881 std::istream\_iterator&lt; \_Tp, \_CharT, \_Traits, \_Dist &gt; Class Template Reference

Inheritance diagram for std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Dist` [difference\\_type](#)
- typedef [basic\\_istream](#)< `_CharT`, `_Traits` > **istream\_type**
- typedef [input\\_iterator\\_tag](#) `iterator_category`
- typedef `const _Tp *` [pointer](#)
- typedef `const _Tp &` [reference](#)
- typedef `_Traits` **traits\_type**
- typedef `_Tp` [value\\_type](#)

## Public Member Functions

- constexpr [istream\\_iterator](#) ()
- [istream\\_iterator](#) ([istream\\_type](#) &\_\_s)
- **istream\_iterator** (const [istream\\_iterator](#) &\_\_obj)
- bool **\_M\_equal** (const [istream\\_iterator](#) &\_\_x) const
- const \_Tp & **operator\*** () const
- [istream\\_iterator](#) & **operator++** ()
- [istream\\_iterator](#) **operator++** (int)
- const \_Tp \* **operator->** () const

### 5.881.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
class std::istream_iterator<_Tp, _CharT, _Traits, _Dist>
```

Provides input iterator semantics for streams.

Definition at line 49 of file `stream_iterator.h`.

### 5.881.2 Member Typedef Documentation

#### 5.881.2.1 `difference_type`

```
typedef _Dist std::iterator< input\_iterator\_tag , _Tp, _Dist , const _Tp * , const _Tp & >↔
::difference\_type [inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

#### 5.881.2.2 `iterator_category`

```
typedef input\_iterator\_tag std::iterator< input\_iterator\_tag , _Tp, _Dist , const _Tp * , const ↵
_Tp & >::iterator\_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

### 5.881.2.3 pointer

```
typedef const _Tp * std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp &
>::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

### 5.881.2.4 reference

```
typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp &
>::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

### 5.881.2.5 value\_type

```
typedef _Tp std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type
[inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

## 5.881.3 Constructor & Destructor Documentation

### 5.881.3.1 istream\_iterator() [1/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
constexpr std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( ) [inline]
```

Construct end of input stream iterator.

Definition at line 64 of file stream\_iterator.h.

### 5.881.3.2 istream\_iterator() [2/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (
    istream_type & __s ) [inline]
```

Construct start of input stream iterator.

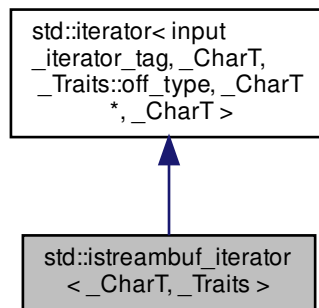
Definition at line 68 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.882 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::istreambuf\_iterator< \_CharT, \_Traits >:



### Public Types

- typedef `_Traits::off_type` [difference\\_type](#)
  - typedef `input_iterator_tag` [iterator\\_category](#)
  - typedef `_CharT *` [pointer](#)
  - typedef `_CharT` [reference](#)
  - typedef `_CharT` [value\\_type](#)
- 
- typedef `_CharT` [char\\_type](#)
  - typedef `_Traits` [traits\\_type](#)
  - typedef `_Traits::int_type` [int\\_type](#)
  - typedef `basic_streambuf< _CharT, _Traits >` [streambuf\\_type](#)
  - typedef `basic_istream< _CharT, _Traits >` [istream\\_type](#)

## Public Member Functions

- constexpr [istreambuf\\_iterator](#) () noexcept
- [istreambuf\\_iterator](#) (const [istreambuf\\_iterator](#) &) noexcept=default
- [istreambuf\\_iterator](#) ([istream\\_type](#) &\_\_s) noexcept
- [istreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- bool [equal](#) (const [istreambuf\\_iterator](#) &\_\_b) const
- [char\\_type](#) [operator\\*](#) () const
- [istreambuf\\_iterator](#) & [operator++](#) ()
- [istreambuf\\_iterator](#) [operator++](#) (int)

## Friends

- template<bool \_IsMove, typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, \_CharT2 \* >::\_\_type [\\_\\_copy\\_move\\_a2](#)  
([istreambuf\\_iterator](#)< \_CharT2 >, [istreambuf\\_iterator](#)< \_CharT2 >, \_CharT2 \*)
- template<typename \_CharT2, typename \_Distance >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, void >::\_\_type [advance](#) ([istreambuf\\_iterator](#)< \_CharT2 > &, \_Distance)
- template<typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, [ostreambuf\\_iterator](#)< \_CharT2 >::\_\_type [copy](#)  
([istreambuf\\_iterator](#)< \_CharT2 >, [istreambuf\\_iterator](#)< \_CharT2 >, [ostreambuf\\_iterator](#)< \_CharT2 >)
- template<typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, [istreambuf\\_iterator](#)< \_CharT2 >::\_\_type [find](#)  
([istreambuf\\_iterator](#)< \_CharT2 >, [istreambuf\\_iterator](#)< \_CharT2 >, const \_CharT2 &)

## 5.882.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >
```

Provides input iterator semantics for streambufs.

Definition at line 125 of file iosfwd.

## 5.882.2 Member Typedef Documentation

## 5.882.2.1 char\_type

```
template<typename _CharT, typename _Traits >
typedef _CharT std::istreambuf\_iterator< _CharT, _Traits >::__char_type
```

Public typedefs.

Definition at line 64 of file streambuf\_iterator.h.



#### 5.882.2.2 difference\_type

```
typedef _Traits::off_type std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT
* , _CharT >::difference_type [inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

#### 5.882.2.3 int\_type

```
template<typename _CharT , typename _Traits >
typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
```

Public typedefs.

Definition at line 66 of file streambuf\_iterator.h.

#### 5.882.2.4 istream\_type

```
template<typename _CharT , typename _Traits >
typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type
```

Public typedefs.

Definition at line 68 of file streambuf\_iterator.h.

#### 5.882.2.5 iterator\_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT , _Traits::off_type , ↵
CharT * , _CharT >::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

#### 5.882.2.6 pointer

```
typedef _CharT * std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , ↵
CharT >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

### 5.882.2.7 reference

```
typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT  
>::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

### 5.882.2.8 streambuf\_type

```
template<typename _CharT , typename _Traits >  
typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::streambuf_type
```

Public typedefs.

Definition at line 67 of file streambuf\_iterator.h.

### 5.882.2.9 traits\_type

```
template<typename _CharT , typename _Traits >  
typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type
```

Public typedefs.

Definition at line 65 of file streambuf\_iterator.h.

### 5.882.2.10 value\_type

```
typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT  
>::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

## 5.882.3 Constructor & Destructor Documentation

**5.882.3.1 istreambuf\_iterator()** [1/3]

```
template<typename _CharT , typename _Traits >
constexpr std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator ( ) [inline], [noexcept]
```

Construct end of input stream iterator.

Definition at line 107 of file streambuf\_iterator.h.

**5.882.3.2 istreambuf\_iterator()** [2/3]

```
template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
    istream_type & __s ) [inline], [noexcept]
```

Construct start of input stream iterator.

Definition at line 117 of file streambuf\_iterator.h.

**5.882.3.3 istreambuf\_iterator()** [3/3]

```
template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
    streambuf_type * __s ) [inline], [noexcept]
```

Construct start of streambuf iterator.

Definition at line 121 of file streambuf\_iterator.h.

**5.882.4 Member Function Documentation****5.882.4.1 equal()**

```
template<typename _CharT , typename _Traits >
bool std::istreambuf_iterator< _CharT, _Traits >::equal (
    const istreambuf_iterator< _CharT, _Traits > & __b ) const [inline]
```

Return true both iterators are end or both are not end.

Definition at line 176 of file streambuf\_iterator.h.

#### 5.882.4.2 `operator*()`

```
template<typename _CharT , typename _Traits >
char_type std::istreambuf_iterator< _CharT, _Traits >::operator* ( ) const [inline]
```

Return the current character pointed to by iterator. This returns `streambuf.sgetc()`. It cannot be assigned. NB: The result of `operator*()` on an end of stream is undefined.

Definition at line 128 of file `streambuf_iterator.h`.

#### 5.882.4.3 `operator++()` [1/2]

```
template<typename _CharT , typename _Traits >
istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits >::operator++ ( ) [inline]
```

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 144 of file `streambuf_iterator.h`.

#### 5.882.4.4 `operator++()` [2/2]

```
template<typename _CharT , typename _Traits >
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (
    int ) [inline]
```

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 158 of file `streambuf_iterator.h`.

The documentation for this class was generated from the following files:

- `iosfwd`
- `streambuf_iterator.h`

### 5.883 `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >` Struct Template Reference

#### Public Types

- typedef `_Distance` `difference_type`
- typedef `_Category` `iterator_category`
- typedef `_Pointer` `pointer`
- typedef `_Reference` `reference`
- typedef `_Tp` `value_type`

### 5.883.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference =  
_Tp&>  
struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 118 of file `stl_iterator_base_types.h`.

### 5.883.2 Member Typedef Documentation

#### 5.883.2.1 `difference_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵  
_Tp*, typename _Reference = _Tp&>  
typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference\_type
```

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

#### 5.883.2.2 `iterator_category`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵  
_Tp*, typename _Reference = _Tp&>  
typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator\_category
```

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

### 5.883.2.3 `pointer`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Pointer std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::pointer
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

### 5.883.2.4 `reference`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Reference std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::reference
```

This type represents a reference-to-value\_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

### 5.883.2.5 `value_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&>
typedef _Tp std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::value\_type
```

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.884 `std::iterator_traits<_Tp*>` Struct Template Reference

### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp\* **pointer**
- typedef \_Tp& **reference**
- typedef \_Tp **value\_type**

#### 5.884.1 Detailed Description

```
template<typename _Tp>
struct std::iterator_traits<_Tp * >
```

Partial specialization for pointer types.

Definition at line 178 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 5.885 `std::iterator_traits< const _Tp * >` Struct Template Reference

##### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

#### 5.885.1 Detailed Description

```
template<typename _Tp>
struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

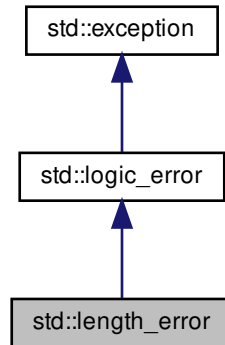
Definition at line 189 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.886 std::length\_error Class Reference

Inheritance diagram for std::length\_error:



### Public Member Functions

- **length\_error** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **length\_error** (const char \*) \_GLIBCXX\_TXN\_SAFE
- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

### 5.886.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a `basic_string` instance).

Definition at line 170 of file `stdexcept`.

### 5.886.2 Member Function Documentation

#### 5.886.2.1 `what()`

```
virtual const char* std::length_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

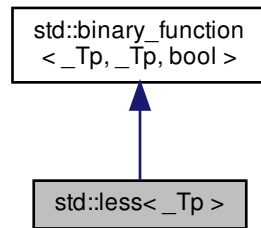
The documentation for this class was generated from the following file:

- [stdexcept](#)



### 5.887 `std::less<_Tp>` Struct Template Reference

Inheritance diagram for `std::less<_Tp>`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR` `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 5.887.1 Detailed Description

```
template<typename _Tp>  
struct std::less<_Tp>
```

One of the [comparison functors](#).

Definition at line 340 of file `stl_function.h`.

#### 5.887.2 Member Typedef Documentation

### 5.887.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

### 5.887.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

### 5.887.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.888 std::less< void > Struct Template Reference

### Public Types

- typedef \_\_is\_transparent **is\_transparent**

### Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t)< std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t)< std::forward< \_Up >(\_\_u))
- template<typename \_Tp, typename \_Up >  
constexpr bool **operator()** (\_Tp \*\_\_t, \_Up \*\_\_u) const noexcept

### 5.888.1 Detailed Description

```
template<>
struct std::less< void >
```

One of the [comparison functors](#).

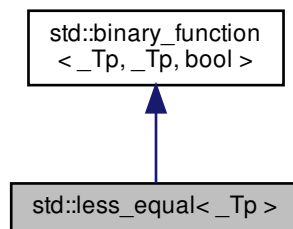
Definition at line 554 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.889 std::less\_equal< \_Tp > Struct Template Reference

Inheritance diagram for `std::less_equal< _Tp >`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.889.1 Detailed Description

```
template<typename _Tp>
struct std::less_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 346 of file `stl_function.h`.

## 5.889.2 Member Typedef Documentation

5.889.2.1 `first_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

5.889.2.2 `result_type`

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

5.889.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

5.890 `std::less_equal< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`)<=`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)<=`std::forward`< `_Up` >(`_u`))
- template<typename `_Tp` , typename `_Up` >  
constexpr bool **operator()** (`_Tp` \*`_t`, `_Up` \*`_u`) const noexcept

### 5.890.1 Detailed Description

```
template<>
struct std::less_equal< void >
```

One of the [comparison functors](#).

Definition at line 678 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.891 `std::linear_congruential_engine< _UIntType, __a, __c, __m >` Class Template Reference

### Public Types

- typedef `_UIntType` [result\\_type](#)

### Public Member Functions

- [linear\\_congruential\\_engine](#) ([result\\_type](#) \_\_s=default\_seed)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type>` [linear\\_congruential\\_engine](#) (`_Sseq` &\_\_q)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator\(\)](#) ()
- void [seed](#) ([result\\_type](#) \_\_s=default\_seed)
- template<typename `_Sseq` > `std::enable_if< std::is_class<_Sseq>::value >::type` [seed](#) (`_Sseq` &\_\_q)

### Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

### Static Public Attributes

- static constexpr [result\\_type](#) [default\\_seed](#)
- static constexpr [result\\_type](#) [increment](#)
- static constexpr [result\\_type](#) [modulus](#)
- static constexpr [result\\_type](#) [multiplier](#)

## Friends

- template<typename \_UIntType1 , \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const  
std::linear\_congruential\_engine< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &\_\_lcr)
- bool operator== (const linear\_congruential\_engine &\_\_lhs, const linear\_congruential\_engine &\_\_rhs)
- template<typename \_UIntType1 , \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits >  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is,  
std::linear\_congruential\_engine< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &\_\_lcr)

## 5.891.1 Detailed Description

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >
```

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 229 of file random.h.

## 5.891.2 Member Typedef Documentation

## 5.891.2.1 result\_type

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

Definition at line 232 of file random.h.

## 5.891.3 Constructor &amp; Destructor Documentation

## 5.891.3.1 linear\_congruential\_engine() [1/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
    result_type __s = default_seed ) [inline], [explicit]
```

Constructs a linear\_congruential\_engine random number generator engine with seed `__s`. The default seed value is 1.

**Parameters**

<code>__s</code>	The initial seed value.
------------------	-------------------------

Definition at line 256 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

**5.891.3.2 linear\_congruential\_engine()** [2/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_
_engine>::value> ::type>
std::linear_congruential_engine<_UIntType, __a, __c, __m >::linear_congruential_engine (
    _Sseq & __q ) [inline], [explicit]
```

Constructs a `linear_congruential_engine` random number generator engine seeded from the seed sequence `__q`.

**Parameters**

<code>__q</code>	the seed sequence.
------------------	--------------------

Definition at line 269 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

**5.891.4 Member Function Documentation****5.891.4.1 discard()**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine<_UIntType, __a, __c, __m >::discard (
    unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Definition at line 313 of file random.h.

## 5.891.4.2 max()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::max ( )
[inline], [static]
```

Gets the largest possible value in the output range.

Definition at line 306 of file random.h.

## 5.891.4.3 min()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::min ( )
[inline], [static]
```

Gets the smallest possible value in the output range.

The minimum depends on the \_\_c parameter: if it is zero, the minimum generated must be > 0, otherwise 0 is allowed.

Definition at line 299 of file random.h.

## 5.891.4.4 operator()()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator() ( ) [inline]
```

Gets the next random number in the sequence.

Definition at line 323 of file random.h.

## 5.891.4.5 seed() [1/2]

```
template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
    result_type __s = default_seed )
```

Reseeds the linear\_congruential\_engine random number generator engine sequence to the seed \_\_s.

## Parameters

<code>__s</code>	The new seed.
------------------	---------------



Seeds the LCR with integral value `__s`, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 117 of file `bits/random.tcc`.

Referenced by `std::linear_congruential_engine<_UIntType, __a, __c, __m>::linear_congruential_engine()`.

#### 5.891.4.6 `seed()` [2/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
std::enable_if< std::is_class< _Sseq >::value >::type std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
    _Sseq & __q )
```

Reseeds the `linear_congruential_engine` random number generator engine sequence using values from the seed sequence `__q`.

##### Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

Seeds the LCR engine with a value generated by `__q`.

Definition at line 133 of file `bits/random.tcc`.

References `std::__lg()`.

### 5.891.5 Friends And Related Function Documentation

#### 5.891.5.1 `operator<<`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr )
[friend]
```

Writes the textual representation of the state `x(i)` of `x` to `__os`.

##### Parameters

<code>__os</code>	The output stream.
<code>__lcr</code>	A % <code>linear_congruential_engine</code> random number generator.

**Returns**`__os.`**5.891.5.2 operator==**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool operator== (
    const linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
    const linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs ) [friend]
```

Compares two linear congruential random number generator objects of the same type for equality.

**Parameters**

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 341 of file random.h.

**5.891.5.3 operator>>**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT
, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr ) [friend]
```

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

**Parameters**

<code>__is</code>	The input stream.
<code>__lcr</code>	A % linear_congruential_engine random number generator.

## Returns

\_\_is.

## 5.891.6 Member Data Documentation

### 5.891.6.1 increment

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr _UIntType std::linear\_congruential\_engine<_UIntType, __a, __c, __m >::increment [static]
```

An increment.

Definition at line 243 of file random.h.

### 5.891.6.2 modulus

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr _UIntType std::linear\_congruential\_engine<_UIntType, __a, __c, __m >::modulus [static]
```

The modulus.

Definition at line 245 of file random.h.

### 5.891.6.3 multiplier

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr _UIntType std::linear\_congruential\_engine<_UIntType, __a, __c, __m >::multiplier [static]
```

The multiplier.

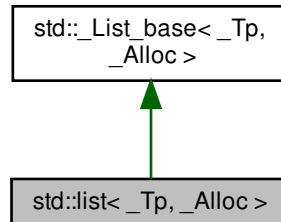
Definition at line 241 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.892 std::list&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::list< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_List_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_traits::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_traits::pointer` **pointer**
- typedef `_Tp_alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `list()`=default
- `list(const allocator_type &__a)` noexcept
- `list(size_type __n, const allocator_type &__a=allocator_type())`
- `list(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `list(const list &__x)`
- `list(list &&)=default`
- `list(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `list(const list &__x, const allocator_type &__a)`
- `list(list &&__x, const allocator_type &__a)` noexcept(`_Node_alloc_traits::_S_always_equal()`)
- template<typename `_InputIterator` , typename `= std::RequireInputIter<_InputIterator>>>`  
`list(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `~list()`=default
- void `assign(size_type __n, const value_type &__val)`

- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (initializer_list< value_type > __l)`
- `reference back () noexcept`
- `const_reference back () const noexcept`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear () noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args>`  
`iterator emplace (const_iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_back (_Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_front (_Args &&... __args)`
- `bool empty () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `iterator erase (const_iterator __position) noexcept`
- `iterator erase (const_iterator __first, const_iterator __last) noexcept`
- `reference front () noexcept`
- `const_reference front () const noexcept`
- `allocator_type get_allocator () const noexcept`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `iterator insert (const_iterator __p, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const value_type &__x)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `size_type max_size () const noexcept`
- `void merge (list && __x)`
- `void merge (list & __x)`
- `template<typename _StrictWeakOrdering >`  
`void merge (list && __x, _StrictWeakOrdering __comp)`
- `template<typename _StrictWeakOrdering >`  
`void merge (list & __x, _StrictWeakOrdering __comp)`
- `list & operator= (const list & __x)`
- `list & operator= (list && __x) noexcept(_Node_alloc_traits::_S_nothrow_move())`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back () noexcept`
- `void pop_front () noexcept`
- `void push_back (const value_type & __x)`
- `void push_back (value_type && __x)`
- `void push_front (const value_type & __x)`
- `void push_front (value_type && __x)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `void remove (const _Tp & __value)`

- template<typename \_Predicate >  
void **remove\_if** (\_Predicate)
- **reverse\_iterator** **rend** () noexcept
- **const\_reverse\_iterator** **rend** () const noexcept
- void **resize** (size\_type \_\_new\_size)
- void **resize** (size\_type \_\_new\_size, const value\_type &\_\_x)
- void **reverse** () noexcept
- size\_type **size** () const noexcept
- void **sort** ()
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering)
- void **splice** (**const\_iterator** \_\_position, **list** &&\_\_x) noexcept
- void **splice** (**const\_iterator** \_\_position, **list** &\_\_x) noexcept
- void **splice** (**const\_iterator** \_\_position, **list** &&\_\_x, **const\_iterator** \_\_i) noexcept
- void **splice** (**const\_iterator** \_\_position, **list** &\_\_x, **const\_iterator** \_\_i) noexcept
- void **splice** (**const\_iterator** \_\_position, **list** &&\_\_x, **const\_iterator** \_\_first, **const\_iterator** \_\_last) noexcept
- void **splice** (**const\_iterator** \_\_position, **list** &\_\_x, **const\_iterator** \_\_first, **const\_iterator** \_\_last) noexcept
- void **swap** (**list** &\_\_x) noexcept
- void **unique** ()
- template<typename \_BinaryPredicate >  
void **unique** (\_BinaryPredicate)

### Protected Types

- typedef **\_List\_node**< \_Tp > **\_Node**

### Protected Member Functions

- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void **\_M\_check\_equal\_allocators** (**list** &\_\_x) noexcept
- void **\_M\_clear** () noexcept
- template<typename... \_Args>  
**\_Node** \* **\_M\_create\_node** (\_Args &&... \_\_args)
- void **\_M\_dec\_size** (size\_t)
- void **\_M\_default\_append** (size\_type \_\_n)
- void **\_M\_default\_initialize** (size\_type \_\_n)
- size\_t **\_M\_distance** (const void \*, const void \*) const
- void **\_M\_erase** (**iterator** \_\_position) noexcept
- void **\_M\_fill\_assign** (size\_type \_\_n, const value\_type &\_\_val)
- void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_x)
- \_Node\_alloc\_traits::pointer **\_M\_get\_node** ()
- \_Node\_alloc\_traits::pointer **\_M\_get\_node** ()
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept

- `size_t _M_get_size () const`
- `void _M_inc_size (size_t)`
- `void _M_init () noexcept`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename... _Args>`  
`void _M_insert (iterator __position, _Args &&... __args)`
- `void _M_move_assign (list &&__x, true_type) noexcept`
- `void _M_move_assign (list &&__x, false_type)`
- `void _M_move_nodes (_List_base &&__x)`
- `size_t _M_node_count () const`
- `void _M_put_node (typename _Node_alloc_traits::pointer __p) noexcept`
- `void _M_put_node (typename _Node_alloc_traits::pointer __p) noexcept`
- `const_iterator _M_resize_pos (size_type &__new_size) const`
- `void _M_set_size (size_t)`
- `void _M_transfer (iterator __position, iterator __first, iterator __last)`

#### Static Protected Member Functions

- `static size_t _S_distance (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last)`
- `static size_t _S_distance (const_iterator, const_iterator)`

#### Protected Attributes

- `_List_impl _M_impl`

#### 5.892.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::list< _Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

#### Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`,

random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

A <--> B <--> C <--> D

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 564 of file `stl_list.h`.

## 5.892.2 Constructor & Destructor Documentation

### 5.892.2.1 list() [1/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list ( ) [default]
```

Creates a list with no elements.

### 5.892.2.2 list() [2/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
    const allocator_type & __a ) [inline], [explicit], [noexcept]
```

Creates a list with no elements.

#### Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 691 of file `stl_list.h`.





**5.892.2.3 list()** [3/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (
    size_type __n,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a list with default constructed elements.

**Parameters**

 <b>__n</b>	The number of elements to initially create.
 <b>__a</b>	An allocator object.

This constructor fills the list with **\_\_n** default constructed elements.

Definition at line 704 of file stl\_list.h.

**5.892.2.4 list()** [4/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type() ) [inline]
```

Creates a list with copies of an exemplar element.

**Parameters**

<b>__n</b>	The number of elements to initially create.
<b>__value</b>	An element to copy.
<b>__a</b>	An allocator object.

This constructor fills the list with **\_\_n** copies of **\_\_value**.

Definition at line 716 of file stl\_list.h.

**5.892.2.5 list()** [5/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (
    const list<_Tp, _Alloc> & __x ) [inline]
```

List copy constructor.

## Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

The newly-created list uses a copy of the allocation object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 743 of file `stl_list.h`.

## 5.892.2.6 list() [6/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
    list< _Tp, _Alloc > && ) [default]
```

List move constructor.

The newly-created list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified list.

## 5.892.2.7 list() [7/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a list from an `initializer_list`.

## Parameters

<code>__l</code>	An <code>initializer_list</code> of <code>value_type</code> .
<code>__a</code>	An allocator object.

Create a list consisting of copies of the elements in the `initializer_list __l`. This is linear in `__l.size()`.

Definition at line 766 of file `stl_list.h`.

## 5.892.2.8 list() [8/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
```

```
std::list< _Tp, _Alloc >::list (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a list from a range.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator object.

Create a list consisting of copies of the elements from `[__first, __last)`. This is linear in N (where N is `distance(__first, __last)`).

Definition at line 811 of file `stl_list.h`.

#### 5.892.2.9 ~list()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::~~list ( ) [default]
```

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### 5.892.3 Member Function Documentation

##### 5.892.3.1 \_M\_create\_node()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
_Node* std::list< _Tp, _Alloc >::_M_create_node (
    _Args &&... __args ) [inline], [protected]
```

#### Parameters

<code>__args</code>	An instance of user data.
---------------------	---------------------------

Allocates space for a new node and constructs a copy of `__args` in it.

Definition at line 640 of file `stl_list.h`.

## 5.892.3.2 assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
    size_type __n,
    const value_type & __val ) [inline]
```

Assigns a given value to a list.

## Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a list with `__n` copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

Definition at line 897 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::operator=()`.

## 5.892.3.3 assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
void std::list< _Tp, _Alloc >::assign (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Assigns a range to a list.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a list with copies of the elements in the range `[__first, __last)`.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

Definition at line 916 of file `stl_list.h`.

**5.892.3.4** `assign()` [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
    initializer_list< value_type > __l ) [inline]
```

Assigns an `initializer_list` to a list.

**Parameters**

↩	An <code>initializer_list</code> of <code>value_type</code> .
↩	
↩	
↩	
/	

Replace the contents of the list with copies of the elements in the `initializer_list __l`. This is linear in `__l.size()`.

Definition at line 938 of file `stl_list.h`.

**5.892.3.5** `back()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::back ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the list.

Definition at line 1138 of file `stl_list.h`.

**5.892.3.6** `back()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::back ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 1150 of file `stl_list.h`.

### 5.892.3.7 begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 953 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::crend(), std::list< \_\_inp, \_\_rebind\_inp >::front(), std::list< \_\_inp, \_\_rebind\_inp >::insert(), std::list< \_\_inp, \_\_rebind\_inp >::list(), std::list< \_\_inp, \_\_rebind\_inp >::merge(), std::operator<(), std::list< \_\_inp, \_\_rebind\_inp >::operator=(), std::operator==( ), std::list< \_\_inp, \_\_rebind\_inp >::pop\_front(), std::list< \_\_inp, \_\_rebind\_inp >::push\_front(), std::list< \_\_inp, \_\_rebind\_inp >::rend(), and std::list< \_\_inp, \_\_rebind\_inp >::splice().

### 5.892.3.8 begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 962 of file stl\_list.h.

### 5.892.3.9 cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 1026 of file stl\_list.h.

### 5.892.3.10 cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 1035 of file stl\_list.h.

#### 5.892.3.11 clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1506 of file `stl_list.h`.

#### 5.892.3.12 crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 1044 of file `stl_list.h`.

#### 5.892.3.13 crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 1053 of file `stl_list.h`.

#### 5.892.3.14 emplace()

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
list< _Tp, _Alloc >::iterator list::emplace (
    const_iterator __position,
    _Args &&... __args )
```

Constructs object in list before specified iterator.

##### Parameters

<code>__position</code>	A <code>const_iterator</code> into the list.
<code>__args</code>	Arguments.

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 90 of file list.tcc.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert().

**5.892.3.15 empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the list is empty. (Thus begin() would equal end().)

Definition at line 1063 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert(), and std::list< \_\_inp, \_\_rebind\_inp >::splice().

**5.892.3.16 end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 971 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::back(), std::list< \_\_inp, \_\_rebind\_inp >::cbegin(), std::list< \_\_inp, \_\_rebind\_inp >::list(), std::list< \_\_inp, \_\_rebind\_inp >::merge(), std::operator<(), std::list< \_\_inp, \_\_rebind\_inp >::operator=(), std::operator==( ), std::list< \_\_inp, \_\_rebind\_inp >::push\_back(), std::list< \_\_inp, \_\_rebind\_inp >::rbegin(), and std::list< \_\_inp, \_\_rebind\_inp >::splice().

**5.892.3.17 end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 980 of file stl\_list.h.

**5.892.3.18 erase()** [1/2]

```
template<typename _Tp, typename _Alloc >
list< _Tp, _Alloc >::iterator list::erase (
    const_iterator __position ) [noexcept]
```

Remove element at given position.



**Parameters**

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

**Returns**

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 152 of file `list.tcc`.

Referenced by `std::list< __inp, __rebind_inp >::erase()`.

**5.892.3.19 erase()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline], [noexcept]
```

Remove a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1464 of file `stl_list.h`.

**5.892.3.20** front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the list.

Definition at line 1122 of file stl\_list.h.

**5.892.3.21** front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 1130 of file stl\_list.h.

**5.892.3.22** get\_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 944 of file stl\_list.h.

**5.892.3.23** insert() [1/5]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc >::iterator list::insert (
    const_iterator __position,
    const value_type & __x )
```

Inserts given value into list before specified iterator.

**Parameters**

<code>__position</code>	A const_iterator into the list.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 103 of file list.tcc.

Referenced by `std::list< __inp, __rebind_inp >::insert()`.

**5.892.3.24 insert()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
    const_iterator __position,
    value_type && __x ) [inline]
```

Inserts given rvalue into list before specified iterator.

**Parameters**

<code>__position</code>	A const_iterator into the list.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1316 of file stl\_list.h.

**5.892.3.25 insert()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
    const_iterator __p,
    initializer_list< value_type > __l ) [inline]
```

Inserts the contents of an initializer\_list into list before specified const\_iterator.

## Parameters

$\leftarrow$ <code>_p</code>	A const_iterator into the list.
$\leftarrow$ <code>_l</code>	An initializer_list of value_type.

## Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert copies of the data in the initializer\_list `l` into the list before the location specified by `p`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1335 of file `stl_list.h`.

## 5.892.3.26 insert() [4/5]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc >::iterator list::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x )
```

Inserts a number of copies of given data into the list.

## Parameters

<code>__position</code>	A const_iterator into the list.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

## Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 118 of file `list.tcc`.

**5.892.3.27 insert()** [5/5]

```
template<typename _Tp , typename _Alloc >
template<typename _InputIterator , typename >
list< _Tp, _Alloc >::iterator list::insert (
    const_iterator __position,
    _InputIterator __first,
    _InputIterator __last )
```

Inserts a range into the list.

**Parameters**

<code>__position</code>	A const_iterator into the list.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

An iterator pointing to the first element inserted (or `__position`).

This function will insert copies of the data in the range *[first,last)* into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 134 of file list.tcc.

**5.892.3.28 max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the size() of the largest possible list.

Definition at line 1073 of file stl\_list.h.

**5.892.3.29 merge()** [1/2]

```
template<typename _Tp , typename _Alloc >
void list::merge (
    list< _Tp, _Alloc > && __x )
```

Merge sorted lists.

## Parameters

<code>__x</code>	Sorted list to merge.
------------------	-----------------------

Assumes that both `__x` and this list are sorted according to operator<(). Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

Definition at line 375 of file list.tcc.

## 5.892.3.30 merge() [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _StrictWeakOrdering >
void list::merge (
    list< _Tp, _Alloc > && __x,
    _StrictWeakOrdering __comp )
```

Merge sorted lists according to comparison function.

## Template Parameters

<code>_StrictWeakOrdering</code>	Comparison function defining sort order.
----------------------------------	--

## Parameters

<code>__x</code>	Sorted list to merge.
<code>__comp</code>	Comparison functor.

Assumes that both `__x` and this list are sorted according to StrictWeakOrdering. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to StrictWeakOrdering().

Definition at line 422 of file list.tcc.

## 5.892.3.31 operator=() [1/3]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc > & list::operator= (
    const list< _Tp, _Alloc > & __x )
```

List assignment operator.

**Parameters**

<code>__l</code>	A list of identical element and allocator types.
<code>__x</code>	

All the elements of `__x` are copied.

Whether the allocator is copied depends on the allocator traits.

Definition at line 268 of file `list.tcc`.

**5.892.3.32 operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (
    list< _Tp, _Alloc > && __x ) [inline], [noexcept]
```

List move assignment operator.

**Parameters**

<code>__l</code>	A list of identical element and allocator types.
<code>__x</code>	

The contents of `__x` are moved into this list (without copying).

Afterwards `__x` is a valid, but unspecified list

Whether the allocator is moved depends on the allocator traits.

Definition at line 861 of file `stl_list.h`.

**5.892.3.33 operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

List initializer list assignment operator.

## Parameters

↵	An initializer_list of value_type.
↵	
↵	
↵	
/	

Replace the contents of the list with copies of the elements in the initializer\_list \_\_l. This is linear in l.size().

Definition at line 879 of file stl\_list.h.

## 5.892.3.34 pop\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back ( ) [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop\_back() is called.

Definition at line 1254 of file stl\_list.h.

## 5.892.3.35 pop\_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_front ( ) [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop\_front() is called.

Definition at line 1205 of file stl\_list.h.

## 5.892.3.36 push\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_back (
    const value_type & __x ) [inline]
```

Add data to the end of the list.



**Parameters**

$\leftrightarrow$ _X	Data to be added.
-------------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1219 of file `stl_list.h`.

**5.892.3.37 push\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_front (
    const value_type & __x ) [inline]
```

Add data to the front of the list.

**Parameters**

$\leftrightarrow$ _X	Data to be added.
-------------------------	-------------------

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1169 of file `stl_list.h`.

**5.892.3.38 rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 989 of file `stl_list.h`.

**5.892.3.39 rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 998 of file `stl_list.h`.

#### 5.892.3.40 remove()

```
template<typename _Tp, typename _Alloc >
void list::remove (
    const _Tp & __value )
```

Remove all elements equal to *value*.

##### Parameters

<code>__value</code>	The value to remove.
----------------------	----------------------

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 326 of file list.tcc.

#### 5.892.3.41 remove\_if()

```
template<typename _Tp , typename _Alloc >
template<typename _Predicate >
void list::remove_if (
    _Predicate __pred )
```

Remove all elements satisfying a predicate.

##### Template Parameters

<code>_Predicate</code>	Unary predicate function or object.
-------------------------	-------------------------------------

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 515 of file list.tcc.

#### 5.892.3.42 rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 1007 of file stl\_list.h.

**5.892.3.43** `rend()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 1016 of file `stl_list.h`.

**5.892.3.44** `resize()` [1/2]

```
template<typename _Tp , typename _Alloc >
void list::resize (
    size_type __new_size )
```

Resizes the list to the specified number of elements.

**Parameters**

<code>__new_size</code>	Number of elements the list should contain.
-------------------------	---

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 231 of file `list.tcc`.

**5.892.3.45** `resize()` [2/2]

```
template<typename _Tp , typename _Alloc >
void list::resize (
    size_type __new_size,
    const value_type & __x )
```

Resizes the list to the specified number of elements.

**Parameters**

<code>__new_size</code>	Number of elements the list should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 243 of file `list.tcc`.

**5.892.3.46 reverse()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::reverse ( ) [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1790 of file stl\_list.h.

**5.892.3.47 size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the number of elements in the list.

Definition at line 1068 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::merge(), and std::operator==().

**5.892.3.48 sort()** [1/2]

```
template<typename _Tp , typename _Alloc >
void list::sort ( )
```

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 468 of file list.tcc.

**5.892.3.49 sort()** [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _StrictWeakOrdering >
void list::sort (
    _StrictWeakOrdering __comp )
```

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 554 of file list.tcc.

**5.892.3.50 splice()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > && __x ) [inline], [noexcept]
```

Insert contents of another list.

**Parameters**

<code>__position</code>	Iterator referencing the element to insert before.
<code>__x</code>	Source list.

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this `!= __x`.

Definition at line 1526 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::splice()`.

**5.892.3.51 splice()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > && __x,
    const_iterator __i ) [inline], [noexcept]
```

Insert element from another list.

**Parameters**

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__i</code>	Const_iterator referencing the element to move.

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1561 of file `stl_list.h`.

**5.892.3.52 splice()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > & __x,
    const_iterator __i ) [inline], [noexcept]
```

Insert element from another list.

## Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__i</code>	Const_iterator referencing the element to move.

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1603 of file `stl_list.h`.

## 5.892.3.53 splice() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > && __x,
    const_iterator __first,
    const_iterator __last ) [inline], [noexcept]
```

Insert range from another list.

## Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__first</code>	Const_iterator referencing the start of range in x.
<code>__last</code>	Const_iterator referencing the end of range in x.

Removes elements in the range `[__first,__last)` and inserts them before `__position` in constant time.

Undefined if `__position` is in `[__first,__last)`.

Definition at line 1622 of file `stl_list.h`.

## 5.892.3.54 splice() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
    const_iterator __position,
    list< _Tp, _Alloc > & __x,
    const_iterator __first,
    const_iterator __last ) [inline], [noexcept]
```

Insert range from another list.

## Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__first</code>	Const_iterator referencing the start of range in x.
<code>__last</code>	Const_iterator referencing the end of range in x.

Removes elements in the range `[__first,__last)` and inserts them before `__position` in constant time.

Undefined if `__position` is in `[__first,__last)`.

Definition at line 1672 of file `stl_list.h`.

5.892.3.55 `swap()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (
    list< _Tp, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another list.

## Parameters

<code>__x</code>	A list of the same element and allocator types.
------------------	---

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1486 of file `stl_list.h`.

5.892.3.56 `unique()` [1/2]

```
template<typename _Tp , typename _Alloc >
void list::unique ( )
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 354 of file `list.tcc`.

## 5.892.3.57 unique() [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _BinaryPredicate >
void list::unique (
    _BinaryPredicate __binary_pred )
```

Remove consecutive elements satisfying a predicate.

## Template Parameters

<code>_BinaryPredicate</code>	Binary predicate function or object.
-------------------------------	--------------------------------------

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 533 of file list.tcc.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 5.893 std::locale Class Reference

## Classes

- class [facet](#)
- class [id](#)

## Public Types

- typedef int [category](#)

## Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &\_\_other) throw ()
- [locale](#) (const char \*\_\_s)
- [locale](#) (const [locale](#) &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const [std::string](#) &\_\_s)
- [locale](#) (const [locale](#) &\_\_base, const [std::string](#) &\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [locale](#) &\_\_add, [category](#) \_\_cat)
- template<typename \_Facet >  
  [locale](#) (const [locale](#) &\_\_other, \_Facet \*\_\_f)



- `~locale () throw ()`
- `template<typename _Facet >  
locale combine (const locale &__other) const`
- `_GLIBCXX_DEFAULT_ABI_TAG string name () const`
- `bool operator!= (const locale &__other) const throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool operator() (const basic_string< _CharT, _Traits, _Alloc > &__s1, const basic_string< _CharT, _Traits, _Alloc > &__s2) const`
- `template<typename _Char, typename _Traits, typename _Alloc >  
bool operator() (const basic_string< _Char, _Traits, _Alloc > &__s1, const basic_string< _Char, _Traits, _Alloc > &__s2) const`
- `const locale & operator= (const locale &__other) throw ()`
- `bool operator== (const locale &__other) const throw ()`

#### Static Public Member Functions

- `static const locale & classic ()`
- `static locale global (const locale &__loc)`

#### Static Public Attributes

- `static const category none`
- `static const category ctype`
- `static const category numeric`
- `static const category collate`
- `static const category time`
- `static const category monetary`
- `static const category messages`
- `static const category all`

#### Friends

- `template<typename _Cache >  
struct __use_cache`
- `class _Impl`
- `class facet`
- `template<typename _Facet >  
bool has_facet (const locale &) throw ()`
- `template<typename _Facet >  
const _Facet & use_facet (const locale &)`

### 5.893.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file locale\_classes.h.

### 5.893.2 Member Typedef Documentation

#### 5.893.2.1 category

```
typedef int std::locale::category
```

Definition of locale::category.

Definition at line 67 of file locale\_classes.h.

### 5.893.3 Constructor & Destructor Documentation

#### 5.893.3.1 locale() [1/8]

```
std::locale::locale ( ) throw ( )
```

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

#### 5.893.3.2 locale() [2/8]

```
std::locale::locale (
    const locale & __other ) throw ( )
```

Copy constructor.

Constructs a copy of *other*.

**Parameters**

<code>__other</code>	The locale to copy.
----------------------	---------------------

**5.893.3.3 locale()** [3/8]

```
std::locale::locale (
    const char * __s ) [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

**Parameters**

<code>__s</code>	Name of the locale to construct.
------------------	----------------------------------

**Exceptions**

<code>std::runtime_error</code>	if <code>__s</code> is null or an undefined locale.
---------------------------------	---

**5.893.3.4 locale()** [4/8]

```
std::locale::locale (
    const locale & __base,
    const char * __s,
    category __cat )
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

**Parameters**

<code>__base</code>	The locale to copy.
<code>__s</code>	Name of the locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from <code>__s</code> .

**Exceptions**

<code>std::runtime_error</code>	if <code>__s</code> is null or an undefined locale.
---------------------------------	---

### 5.893.3.5 locale() [5/8]

```
std::locale::locale (
    const std::string & __s ) [inline], [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

#### Parameters

<code>__s</code>	Name of the locale to construct.
------------------	----------------------------------

#### Exceptions

<code>std::runtime_error</code>	if <code>__s</code> is an undefined locale.
---------------------------------	---

Definition at line 163 of file locale\_classes.h.

### 5.893.3.6 locale() [6/8]

```
std::locale::locale (
    const locale & __base,
    const std::string & __s,
    category __cat ) [inline]
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

#### Parameters

<code>__base</code>	The locale to copy.
<code>__s</code>	Name of the locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from <code>__s</code> .

#### Exceptions

<code>std::runtime_error</code>	if <code>__s</code> is an undefined locale.
---------------------------------	---

Definition at line 177 of file locale\_classes.h.

**5.893.3.7 locale()** [7/8]

```
std::locale::locale (
    const locale & __base,
    const locale & __add,
    category __cat )
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

**Parameters**

<code>__base</code>	The locale to copy.
<code>__add</code>	The locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from add.

**5.893.3.8 locale()** [8/8]

```
template<typename _Facet >
std::locale::locale (
    const locale & __other,
    _Facet * __f )
```

Construct locale with another facet.

Constructs a copy of the locale *\_\_other*. The facet *\_\_f* is added to *\_\_other*, replacing an existing facet of type *Facet* if there is one. If *\_\_f* is null, this locale is a copy of *\_\_other*.

**Parameters**

<code>__other</code>	The locale to copy.
<code>__f</code>	The facet to add in.

Definition at line 45 of file `locale_classes.tcc`.

**5.893.3.9 ~locale()**

```
std::locale::~~locale ( ) throw ( )
```

Locale destructor.

## 5.893.4 Member Function Documentation

## 5.893.4.1 classic()

```
static const locale& std::locale::classic ( ) [static]
```

Return reference to the C locale.

## 5.893.4.2 combine()

```
template<typename _Facet >  
locale std::locale::combine (   
    const locale & __other ) const
```

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type `Facet` from the locale *other* into the new locale.

## Template Parameters

<code>_Facet</code>	The facet type to copy from other
---------------------	-----------------------------------

## Parameters

<code>__other</code>	The locale to copy from.
----------------------	--------------------------

## Returns

Newly constructed locale.

## Exceptions

<code>std::runtime_error</code>	if <code>__other</code> has no facet of type <code>_Facet</code> .
---------------------------------	--

Definition at line 63 of file `locale_classes.tcc`.

## 5.893.4.3 global()

```
static locale std::locale::global (   
    const locale & __loc ) [static]
```

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

#### Parameters

<code>__loc</code>	The new locale to make global.
--------------------	--------------------------------

#### Returns

Copy of the old global locale.

#### 5.893.4.4 `name()`

```
_GLIBCXX_DEFAULT_ABI_TAG string std::locale::name ( ) const
```

Return locale name.

#### Returns

Locale name or "\*" if unnamed.

#### 5.893.4.5 `operator!=( )`

```
bool std::locale::operator!= (
    const locale & __other ) const throw ( )    [inline]
```

Locale inequality.

#### Parameters

<code>__other</code>	The locale to compare against.
----------------------	--------------------------------

#### Returns

`! (*this == __other)`

Definition at line 264 of file `locale_classes.h`.

## 5.893.4.6 operator()()

```
template<typename _Char , typename _Traits , typename _Alloc >
bool std::locale::operator() (
    const basic_string< _Char, _Traits, _Alloc > & __s1,
    const basic_string< _Char, _Traits, _Alloc > & __s2 ) const
```

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

## Parameters

<code>__s1</code>	First string to compare.
<code>__s2</code>	Second string to compare.

## Returns

True if `collate<_Char>` facet compares `__s1 < __s2`, else false.

## 5.893.4.7 operator=( )

```
const locale& std::locale::operator= (
    const locale & __other ) throw ( )
```

Assignment operator.

Set this locale to be a copy of *other*.

## Parameters

<code>__other</code>	The locale to copy.
----------------------	---------------------

## Returns

A reference to this locale.

## 5.893.4.8 operator==( )

```
bool std::locale::operator== (
    const locale & __other ) const throw ( )
```

Locale equality.



**Parameters**

<code>__other</code>	The locale to compare against.
----------------------	--------------------------------

**Returns**

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

**5.893.5 Friends And Related Function Documentation****5.893.5.1 has\_facet**

```
template<typename _Facet >
bool has_facet (
    const locale & ) throw ( ) [friend]
```

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Template Parameters**

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

**Parameters**

<code>__loc</code>	The locale to test.
--------------------	---------------------

**Returns**

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file locale\_classes.tcc.

**5.893.5.2 use\_facet**

```
template<typename _Facet >
const _Facet& use_facet (
    const locale & ) [friend]
```

Return a facet.

use\_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has\_facet(locale) is true, there is a suitable facet to return. It throws std::bad\_cast if the locale doesn't contain a facet of type Facet.

## Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

## Parameters

<code>__loc</code>	The locale to use.
--------------------	--------------------

## Returns

Reference to facet of type Facet.

## Exceptions

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
----------------------------	---

Definition at line 132 of file `locale_classes.tcc`.

## 5.893.6 Member Data Documentation

## 5.893.6.1 all

```
const category std::locale::all [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

## 5.893.6.2 collate

```
const category std::locale::collate [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

#### 5.893.6.3 ctype

```
const category std::locale::ctype [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

#### 5.893.6.4 messages

```
const category std::locale::messages [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

#### 5.893.6.5 monetary

```
const category std::locale::monetary [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

#### 5.893.6.6 none

```
const category std::locale::none [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

#### 5.893.6.7 numeric

```
const category std::locale::numeric [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

#### 5.893.6.8 time

```
const category std::locale::time [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

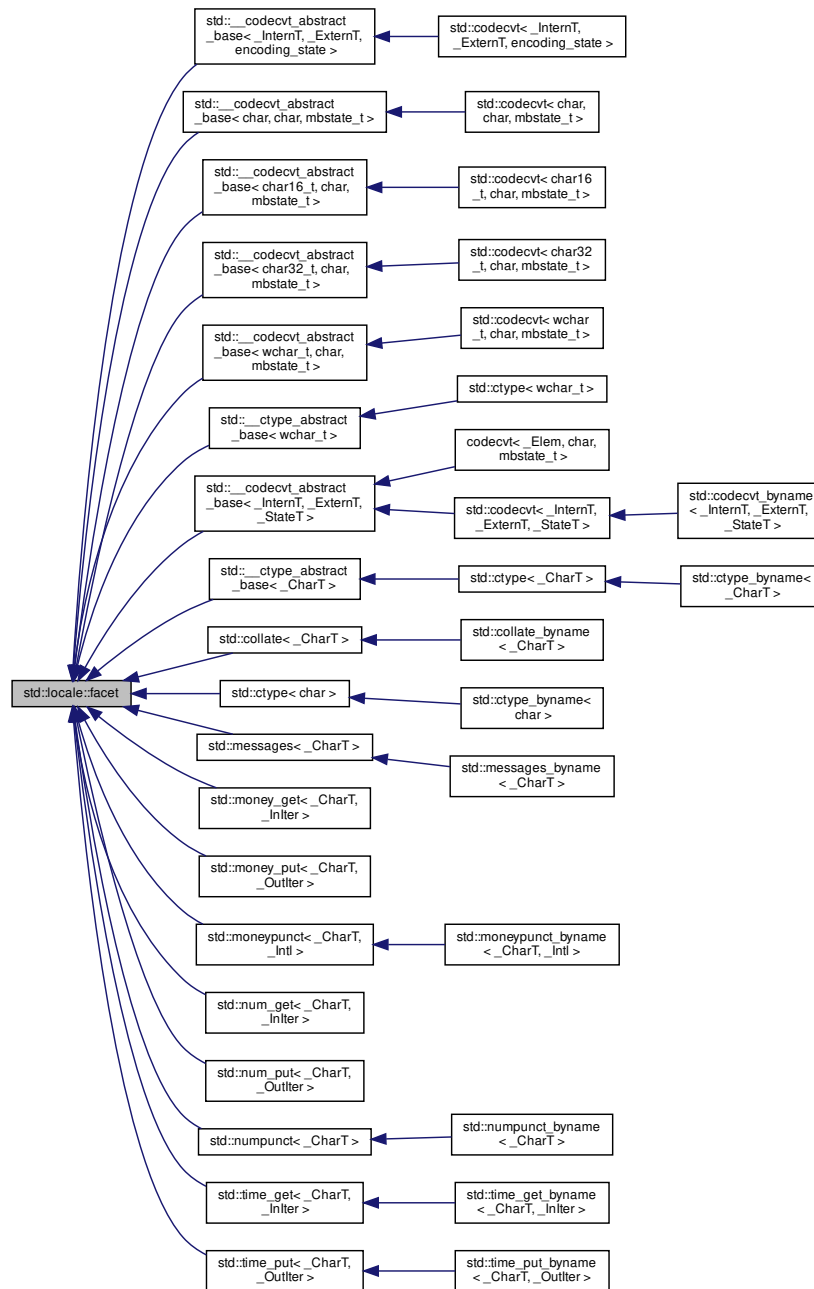
Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 5.894 std::locale::facet Class Reference

Inheritance diagram for std::locale::facet:



### Protected Member Functions

- [facet](#) (size\_t \_\_refs=0) throw ()

- **facet** (const **facet** &)=delete
- virtual **~facet** ()
- **facet** & **operator=** (const **facet** &)=delete

#### Static Protected Member Functions

- static **\_\_c\_locale** **\_S\_clone\_c\_locale** (**\_\_c\_locale** &**\_\_cloc**) throw ()
- static void **\_S\_create\_c\_locale** (**\_\_c\_locale** &**\_\_cloc**, const char \***\_\_s**, **\_\_c\_locale** **\_\_old**=0)
- static void **\_S\_destroy\_c\_locale** (**\_\_c\_locale** &**\_\_cloc**)
- static **\_\_c\_locale** **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static **\_\_c\_locale** **\_S\_lc\_ctype\_c\_locale** (**\_\_c\_locale** **\_\_cloc**, const char \***\_\_s**)

#### Friends

- class **locale**
- class **locale::\_Impl**

#### 5.894.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

Definition at line 371 of file locale\_classes.h.

#### 5.894.2 Constructor & Destructor Documentation

##### 5.894.2.1 facet()

```
std::locale::facet::facet (
    size_t __refs = 0 ) throw ( )    [inline], [explicit], [protected]
```

Facet constructor.

This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

#### Parameters

<b>__refs</b>	The initial value for reference count.
---------------	--

Definition at line 403 of file locale\_classes.h.

#### 5.894.2.2 ~facet()

```
virtual std::locale::facet::~~facet ( ) [protected], [virtual]
```

Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

### 5.895 std::locale::id Class Reference

#### Public Member Functions

- [id](#) ()
- `size_t __M_id () const` throw ()

#### Friends

- `template<typename _Facet >`  
`bool has\_facet (const locale &) throw ()`
- class **locale**
- class **locale::\_Impl**
- `template<typename _Facet >`  
`const _Facet & use\_facet (const locale &)`

#### 5.895.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Definition at line 483 of file locale\_classes.h.

#### 5.895.2 Constructor & Destructor Documentation

## 5.895.2.1 id()

```
std::locale::id::id ( ) [inline]
```

Constructor.

Definition at line 514 of file locale\_classes.h.

## 5.895.3 Friends And Related Function Documentation

## 5.895.3.1 has\_facet

```
template<typename _Facet >
bool has_facet (
    const locale & ) throw ( ) [friend]
```

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

## Template Parameters

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

## Parameters

<code>__loc</code>	The locale to test.
--------------------	---------------------

## Returns

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file locale\_classes.tcc.

## 5.895.3.2 use\_facet

```
template<typename _Facet >
const _Facet& use_facet (
    const locale & ) [friend]
```

Return a facet.

use\_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has\_↔  
facet(locale) is true, there is a suitable facet to return. It throws std::bad\_cast if the locale doesn't contain a facet of type Facet.



### Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

### Parameters

<code>__loc</code>	The locale to use.
--------------------	--------------------

### Returns

Reference to facet of type Facet.

### Exceptions

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
----------------------------	---

Definition at line 132 of file `locale_classes.tcc`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.896 `std::lock_guard<_Mutex>` Class Template Reference

### Public Types

- typedef `_Mutex` **`mutex_type`**

### Public Member Functions

- **`lock_guard`** (`mutex_type &__m`)
- **`lock_guard`** (`mutex_type &__m`, [adopt\\_lock\\_t](#)) noexcept
- **`lock_guard`** (const [lock\\_guard](#) &)=delete
- [lock\\_guard](#) & **`operator=`** (const [lock\\_guard](#) &)=delete

### 5.896.1 Detailed Description

```
template<typename _Mutex>
class std::lock_guard<_Mutex >
```

A simple scoped lock type.

A `lock_guard` controls mutex ownership within a scope, releasing ownership in the destructor.

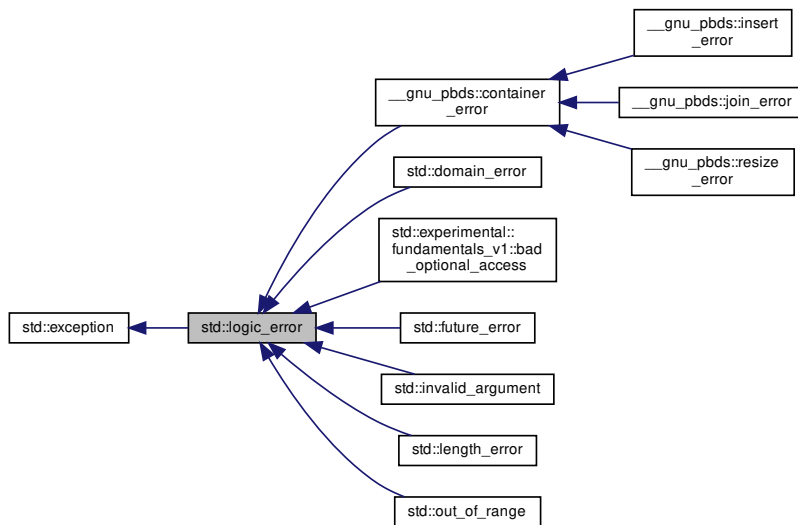
Definition at line 156 of file `std_mutex.h`.

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

## 5.897 std::logic\_error Class Reference

Inheritance diagram for std::logic\_error:



## Public Member Functions

- [logic\\_error](#) (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **logic\_error** (const char \*) \_GLIBCXX\_TXN\_SAFE
- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.897.1 Detailed Description

One of two subclasses of exception.

Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 113 of file stdexcept.

## 5.897.2 Constructor &amp; Destructor Documentation

### 5.897.2.1 `logic_error()`

```
std::logic_error::logic_error (
    const string & __arg ) [explicit]
```

Takes a character string describing the error.

## 5.897.3 Member Function Documentation

### 5.897.3.1 `what()`

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

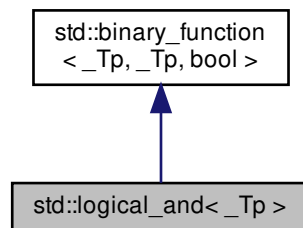
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.898 `std::logical_and<_Tp>` Struct Template Reference

Inheritance diagram for `std::logical_and<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `_GLIBCXX14_CONSTEXPR bool operator() (const _Tp &__x, const _Tp &__y) const`

### 5.898.1 Detailed Description

```
template<typename _Tp>
struct std::logical_and<_Tp>
```

One of the [Boolean operations functors](#).

Definition at line 751 of file `stl_function.h`.

### 5.898.2 Member Typedef Documentation

#### 5.898.2.1 `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.898.2.2 `result_type`

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 5.898.2.3 `second_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.899 `std::logical_and< void >` Struct Template Reference

### Public Types

- typedef `__is_transparent` **is\_transparent**

### Public Member Functions

- template<typename `_Tp`, typename `_Up` >  
`_GLIBCXX14_CONSTEXPR` auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`<  
`_Tp` >(`_t`) &&`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`) &&`std::forward`< `_Up` >(`_u`))

### 5.899.1 Detailed Description

```
template<>
struct std::logical_and< void >
```

One of the [Boolean operations functors](#).

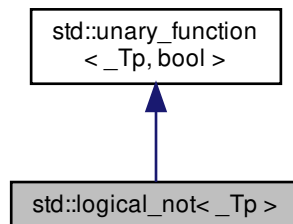
Definition at line 793 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.900 `std::logical_not< _Tp >` Struct Template Reference

Inheritance diagram for `std::logical_not< _Tp >`:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

### Public Member Functions

- `_GLIBCXX14_CONSTEXPR` `bool` **operator()** (`const _Tp &__x`) `const`

#### 5.900.1 Detailed Description

```
template<typename _Tp>  
struct std::logical_not<_Tp>
```

One of the [Boolean operations functors](#).

Definition at line 757 of file `stl_function.h`.

#### 5.900.2 Member Typedef Documentation

##### 5.900.2.1 `argument_type`

```
typedef _Tp std::unary\_function<_Tp, bool>::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

##### 5.900.2.2 `result_type`

```
typedef bool std::unary\_function<_Tp, bool>::result\_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.901 `std::logical_not< void >` Struct Template Reference

### Public Types

- typedef `__is_transparent` **is\_transparent**

### Public Member Functions

- template<typename `_Tp` >  
    \_GLIBCXX14\_CONSTEXPR auto **operator()** (`_Tp` &&\_\_t) const noexcept(noexcept(!std::forward< `_Tp` >(\_\_t)))  
    -> decltype(!std::forward< `_Tp` >(\_\_t))

### 5.901.1 Detailed Description

```
template<>
struct std::logical_not< void >
```

One of the [Boolean operations functors](#).

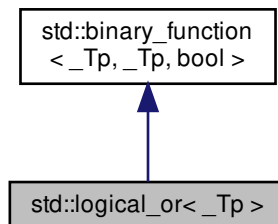
Definition at line 823 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.902 `std::logical_or< _Tp >` Struct Template Reference

Inheritance diagram for `std::logical_or< _Tp >`:



### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- \_GLIBCXX14\_CONSTEXPR bool **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

#### 5.902.1 Detailed Description

```
template<typename _Tp>  
struct std::logical_or< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 754 of file stl\_function.h.

#### 5.902.2 Member Typedef Documentation

##### 5.902.2.1 first\_argument\_type

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::first\_argument\_type [inherited]
```

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 121 of file stl\_function.h.

##### 5.902.2.2 result\_type

```
typedef bool std::binary\_function< _Tp , _Tp , bool >::result\_type [inherited]
```

[result\\_type](#) is the return type

Definition at line 127 of file stl\_function.h.



### 5.902.2.3 `second_argument_type`

```
typedef _Tp std::binary\_function< _Tp , _Tp , bool >::second\_argument\_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.903 `std::logical_or< void >` Struct Template Reference

### Public Types

- typedef `__is_transparent` **`is_transparent`**

### Public Member Functions

- `template<typename _Tp , typename _Up >`  
`_GLIBCXX14_CONSTEXPR auto operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward<  
_Tp >(__t)||std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)||std::forward< _Up >(__u))`

### 5.903.1 Detailed Description

```
template<>  
struct std::logical\_or< void >
```

One of the [Boolean operations functors](#).

Definition at line 808 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.904 `std::lognormal_distribution< _RealType >` Class Template Reference

### Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **lognormal\_distribution** (\_RealType \_\_m=\_RealType(0), \_RealType \_\_s=\_RealType(1))
- **lognormal\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- \_RealType **s** () const

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::lognormal\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [lognormal\\_distribution](#) &\_\_d1, const [lognormal\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::lognormal\\_distribution](#)< \_RealType1 > &\_\_x)

## 5.904.1 Detailed Description

```
template<typename _RealType = double>
class std::lognormal_distribution<_RealType>
```

A [lognormal\\_distribution](#) random number distribution.

The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2143 of file random.h.

## 5.904.2 Member Typedef Documentation

### 5.904.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::lognormal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2146 of file random.h.

## 5.904.3 Member Function Documentation

### 5.904.3.1 max()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2239 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

### 5.904.3.2 min()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2232 of file random.h.

### 5.904.3.3 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::lognormal_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 2247 of file random.h.

## 5.904.3.4 param() [1/2]

```
template<typename _RealType = double>
param_type std::lognormal_distribution<_RealType>::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2217 of file random.h.

## 5.904.3.5 param() [2/2]

```
template<typename _RealType = double>
void std::lognormal_distribution<_RealType>::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2225 of file random.h.

## 5.904.3.6 reset()

```
template<typename _RealType = double>
void std::lognormal_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 2199 of file random.h.

References `std::normal_distribution<_RealType>::reset()`.

## 5.904.4 Friends And Related Function Documentation

## 5.904.4.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::lognormal_distribution<_RealType1> & __x ) [friend]
```

Inserts a `lognormal_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A lognormal_distribution random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**5.904.4.2 operator==**

```
template<typename _RealType = double>
bool operator== (
    const lognormal_distribution< _RealType > & __d1,
    const lognormal_distribution< _RealType > & __d2 ) [friend]
```

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2284 of file random.h.

**5.904.4.3 operator>>**

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::lognormal_distribution< _RealType1 > & __x ) [friend]
```

Extracts a lognormal\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A lognormal_distribution random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.905 `std::lognormal_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [lognormal\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- param\_type** (\_RealType \_\_m=\_RealType(0), \_RealType \_\_s=\_RealType(1))
- \_RealType **m** () const
- \_RealType **s** () const

### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.905.1 Detailed Description

```
template<typename _RealType = double>
struct std::lognormal_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2153 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.906 `std::make_signed<_Tp>` Struct Template Reference

### Public Types

- typedef \_\_make\_signed\_selector<\_Tp>::\_\_type **type**

#### 5.906.1 Detailed Description

```
template<typename _Tp>
struct std::make_signed<_Tp>
```

`make_signed`

Definition at line 1675 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.907 `std::make_unsigned< _Tp >` Struct Template Reference

### Public Types

- `typedef __make_unsigned_selector< _Tp >::__type type`

### 5.907.1 Detailed Description

```
template<typename _Tp>
struct std::make_unsigned< _Tp >
```

`make_unsigned`

Definition at line 1586 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.908 `std::map< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

### Public Types

- `typedef _Alloc allocator_type`
- `typedef _Rep_type::const_iterator const_iterator`
- `typedef _Alloc_traits::const_pointer const_pointer`
- `typedef _Alloc_traits::const_reference const_reference`
- `typedef \_Rep\_type::const\_reverse\_iterator const_reverse_iterator`
- `typedef _Rep_type::difference_type difference_type`
- `typedef _Rep_type::iterator iterator`
- `typedef _Compare key_compare`
- `typedef _Key key_type`
- `typedef _Tp mapped_type`
- `typedef _Alloc_traits::pointer pointer`
- `typedef _Alloc_traits::reference reference`
- `typedef \_Rep\_type::reverse\_iterator reverse_iterator`
- `typedef _Rep_type::size_type size_type`
- `typedef std::pair< const _Key, _Tp > value_type`

## Public Member Functions

- [map](#) ()=default
- [map](#) (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- [map](#) (const [map](#) &)=default
- [map](#) ([map](#) &&)=default
- [map](#) (initializer\_list< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- [map](#) (const allocator\_type &\_\_a)
- [map](#) (const [map](#) &\_\_m, const allocator\_type &\_\_a)
- [map](#) ([map](#) &&\_\_m, const allocator\_type &\_\_a) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value && \_Alloc\_traits::S\_always\_equal())
- [map](#) (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
[map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
[map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator >  
[map](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- [~map](#) ()=default
- mapped\_type & [at](#) (const key\_type &\_\_k)
- const mapped\_type & [at](#) (const key\_type &\_\_k) const
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > [emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator [emplace\\_hint](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const\_iterator [end](#) () const noexcept
- size\_type [erase](#) (const key\_type &\_\_x)
- iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- allocator\_type [get\\_allocator](#) () const noexcept
- void [insert](#) (std::initializer\_list< value\_type > \_\_list)
- template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- key\_compare [key\\_comp](#) () const
- size\_type [max\\_size](#) () const noexcept
- [map](#) & [operator=](#) (const [map](#) &)=default
- [map](#) & [operator=](#) ([map](#) &&)=default
- [map](#) & [operator=](#) (initializer\_list< value\_type > \_\_l)
- mapped\_type & [operator\[\]](#) (const key\_type &\_\_k)
- mapped\_type & [operator\[\]](#) (key\_type &&\_\_k)
- reverse\_iterator [rbegin](#) () noexcept



- `const_reverse_iterator rbegin` () const noexcept
- `reverse_iterator rend` () noexcept
- `const_reverse_iterator rend` () const noexcept
- `size_type size` () const noexcept
- `void swap` (`map` &\_\_x) noexcept(*/\*conditional \*/*)
- `value_compare value_comp` () const

- `std::pair`< iterator, bool > `insert` (const `value_type` &\_\_x)
- `std::pair`< iterator, bool > `insert` (`value_type` &&\_\_x)
- `template`<typename `_Pair` >  
    \_\_enable\_if\_t< `is_constructible`< `value_type`, `_Pair` >::value, `pair`< iterator, bool > > `insert` (`_Pair` &&\_\_x)

- iterator `insert` (const\_iterator \_\_position, const `value_type` &\_\_x)
- iterator `insert` (const\_iterator \_\_position, `value_type` &&\_\_x)
- `template`<typename `_Pair` >  
    \_\_enable\_if\_t< `is_constructible`< `value_type`, `_Pair` >::value, iterator > `insert` (const\_iterator \_\_position, `_Pair` &&\_\_x)

- iterator `erase` (const\_iterator \_\_position)
- `_GLIBCXX_ABI_TAG_CXX11` iterator `erase` (iterator \_\_position)

- iterator `find` (const key\_type &\_\_x)
- `template`<typename `_Kt` >  
    auto `find` (const `_Kt` &\_\_x) -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))

- const\_iterator `find` (const key\_type &\_\_x) const
- `template`<typename `_Kt` >  
    auto `find` (const `_Kt` &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))

- `size_type count` (const key\_type &\_\_x) const
- `template`<typename `_Kt` >  
    auto `count` (const `_Kt` &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))

- iterator [lower\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt >  
auto [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt >  
auto [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt >  
auto [upper\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt >  
auto [upper\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- std::pair< iterator, iterator > [equal\\_range](#) (const key\_type &\_\_x)
- template<typename \_Kt >  
auto [equal\\_range](#) (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))
- std::pair< const\_iterator, const\_iterator > [equal\\_range](#) (const key\_type &\_\_x) const
- template<typename \_Kt >  
auto [equal\\_range](#) (const \_Kt &\_\_x) const -> decltype(pair< const\_iterator, const\_iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))

#### Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**< (const [map](#)< \_K1, \_T1, \_C1, \_A1 > &, const [map](#)< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**== (const [map](#)< \_K1, \_T1, \_C1, \_A1 > &, const [map](#)< \_K1, \_T1, \_C1, \_A1 > &)

#### 5.908.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>>
class std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

**Template Parameters**

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 100 of file `stl_map.h`.

**5.908.2 Constructor & Destructor Documentation****5.908.2.1 `map()`** [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map ( ) [default]
```

Default constructor creates no elements.

**5.908.2.2 `map()`** [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a map with no elements.

**Parameters**

<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Definition at line 192 of file stl\_map.h.

### 5.908.2.3 map() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    const map< _Key, _Tp, _Compare, _Alloc > & ) [default]
```

Map copy constructor.

Whether the allocator is copied depends on the allocator traits.

### 5.908.2.4 map() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    map< _Key, _Tp, _Compare, _Alloc > && ) [default]
```

Map move constructor.

The newly-created map contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, map.

### 5.908.2.5 map() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    initializer_list< value_type > __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a map from an initializer\_list.

#### Parameters

<code>__l</code>	An initializer_list.
<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Create a map consisting of copies of the elements in the initializer\_list `__l`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 226 of file stl\_map.h.

**5.908.2.6 map()** [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    const allocator_type & __a ) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 234 of file stl\_map.h.

**5.908.2.7 map()** [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    const map< _Key, _Tp, _Compare, _Alloc > & __m,
    const allocator_type & __a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 238 of file stl\_map.h.

**5.908.2.8 map()** [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    map< _Key, _Tp, _Compare, _Alloc > && __m,
    const allocator_type & __a ) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 242 of file stl\_map.h.

**5.908.2.9 map()** [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    initializer_list< value_type > __l,
    const allocator_type & __a ) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 248 of file stl\_map.h.

## 5.908.2.10 map() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a ) [inline]
```

Allocator-extended range constructor.

Definition at line 254 of file stl\_map.h.

## 5.908.2.11 map() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Builds a map from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 271 of file stl\_map.h.

## 5.908.2.12 map() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::map< _Key, _Tp, _Compare, _Alloc >::map (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a map from a range.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a map consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 288 of file `stl_map.h`.

**5.908.2.13 ~map()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::~~map ( ) [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**5.908.3 Member Function Documentation****5.908.3.1 at()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type& std::map<_Key, _Tp, _Compare, _Alloc >::at (
    const key_type & __k ) [inline]
```

Access to map data.

**Parameters**

<code>__k</code>	The key for which data should be retrieved.
------------------	---

**Returns**

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

## Exceptions

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 535 of file stl\_map.h.

References `std::map< _Key, _Tp, _Compare, _Alloc >::end()`, `std::map< _Key, _Tp, _Compare, _Alloc >::key_comp()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound()`.

## 5.908.3.2 begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 354 of file stl\_map.h.

## 5.908.3.3 begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 363 of file stl\_map.h.

## 5.908.3.4 cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 427 of file stl\_map.h.



#### 5.908.3.5 cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 436 of file `stl_map.h`.

#### 5.908.3.6 clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map<_Key, _Tp, _Compare, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1132 of file `stl_map.h`.

#### 5.908.3.7 count() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map<_Key, _Tp, _Compare, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements with given key.

##### Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

##### Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 1214 of file `stl_map.h`.

**5.908.3.8 count()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::count (
    const _Kt & __x ) const -> decltype(_M_t._M_count_tr(__x))    [inline]
```

Finds the number of elements with given key.

**Parameters**

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

**Returns**

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 1220 of file stl\_map.h.

**5.908.3.9 crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin ( ) const    [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 445 of file stl\_map.h.

**5.908.3.10 crend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crend ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 454 of file stl\_map.h.

**5.908.3.11** `emplace()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >>  
template<typename... _Args>  
std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc >::emplace (  
    _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the map.

## Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 574 of file `stl_map.h`.

5.908.3.12 `emplace_hint()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the map.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↔associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↔associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 604 of file `stl_map.h`.

**5.908.3.13 empty()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map<_Key, _Tp, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the map is empty. (Thus begin() would equal end().)

Definition at line 463 of file stl\_map.h.

**5.908.3.14 end()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 372 of file stl\_map.h.

Referenced by std::map<\_Key, \_Tp, \_Compare, \_Alloc >::at().

**5.908.3.15 end()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 381 of file stl\_map.h.

**5.908.3.16 equal\_range()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, iterator> std::map<_Key, _Tp, _Compare, _Alloc >::equal_range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>_↔</code>	Key of (key, value) pairs to be located.
<code>_X</code>	

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1332 of file stl\_map.h.

## 5.908.3.17 equal\_range() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

## Parameters

<code>_↔</code>	Key of (key, value) pairs to be located.
<code>_X</code>	

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1338 of file stl\_map.h.

**5.908.3.18** `equal_range()` [ 3 / 4 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<const_iterator, const_iterator> std::map<_Key, _Tp, _Compare, _Alloc >::equal_range (
    const key_type & __x ) const    [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1361 of file `stl_map.h`.

**5.908.3.19** `equal_range()` [ 4 / 4 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map<_Key, _Tp, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) const -> decltype(pair<const_iterator, const_iterator>(_M_t._M_
equal_range_tr(__x)))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1367 of file stl\_map.h.

#### 5.908.3.20 erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
    const_iterator __position ) [inline]
```

Erases an element from a map.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

##### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1030 of file stl\_map.h.

#### 5.908.3.21 erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
_GLIBCXX_ABI_TAG_CXX11 iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
    iterator __position ) [inline]
```

Erases an element from a map.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---



**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1036 of file `stl_map.h`.

**5.908.3.22 erase()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

**Parameters**

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1067 of file `stl_map.h`.

**5.908.3.23 erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a `[first,last)` range of elements from a map.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

## Returns

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1087 of file `stl_map.h`.

## 5.908.3.24 find() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in a map.

## Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1168 of file `stl_map.h`.

## 5.908.3.25 find() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x ) -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a map.

**Parameters**

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1174 of file `stl_map.h`.

**5.908.3.26 find()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in a map.

**Parameters**

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

**Returns**

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1193 of file `stl_map.h`.

**5.908.3.27 find()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x ) const -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a map.

## Parameters

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

## Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1199 of file `stl_map.h`.

## 5.908.3.28 get\_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 344 of file `stl_map.h`.

## 5.908.3.29 insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, bool> std::map< _Key, _Tp, _Compare, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Attempts to insert a `std::pair` into the map.

## Parameters

<code>_↵</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
<code>_X</code>	

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 801 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::insert()`.

#### 5.908.3.30 `insert()` [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc >::insert (
    value_type && __x ) [inline]
```

Attempts to insert a `std::pair` into the map.

##### Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

##### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 808 of file `stl_map.h`.

#### 5.908.3.31 `insert()` [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair>::value, pair<iterator, bool> > std::map<__↵
_Key, _Tp, _Compare, _Alloc >::insert (
    _Pair && __x ) [inline]
```

Attempts to insert a `std::pair` into the map.

## Parameters

<code>_↵</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
<code>_X</code>	

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 814 of file `stl_map.h`.

## 5.908.3.32 insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
    std::initializer_list< value_type > __list ) [inline]
```

Attempts to insert a list of `std::pairs` into the map.

## Parameters

<code>__list</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted.
---------------------	---

Complexity similar to that of the range constructor.

Definition at line 828 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

## 5.908.3.33 insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x ) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 858 of file `stl_map.h`.

**5.908.3.34 insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    value_type && __x ) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 868 of file `stl_map.h`.

## 5.908.3.35 insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair>::value, iterator> std::map< _Key, _Tp, _↵
_Compare, _Alloc >::insert (
    const_iterator __position,
    _Pair && __x ) [inline]
```

Attempts to insert a std::pair into the map.

## Parameters

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 873 of file stl\_map.h.

## 5.908.3.36 insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Template function that attempts to insert a range of elements.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.



Complexity similar to that of the range constructor.

Definition at line 891 of file `stl_map.h`.

#### 5.908.3.37 `key_comp()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::map< _Key, _Tp, _Compare, _Alloc >::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 1141 of file `stl_map.h`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::at()`.

#### 5.908.3.38 `lower_bound()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

##### Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

##### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 1238 of file `stl_map.h`.



Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::at()`.

## 5.908.3.39 lower\_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))    [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

	Key of (key, value) pair to be located.
	

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.



Definition at line 1244 of file stl\_map.h.

## 5.908.3.40 lower\_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) const    [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

	Key of (key, value) pair to be located.
	

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 1263 of file stl\_map.h.

**5.908.3.41** `lower_bound()` [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 1269 of file `stl_map.h`.

**5.908.3.42** `max_size()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the maximum size of the map.

Definition at line 473 of file `stl_map.h`.

**5.908.3.43** `operator=()` [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
    const map< _Key, _Tp, _Compare, _Alloc > & ) [default]
```

Map assignment operator.

Whether the allocator is copied depends on the allocator traits.

**5.908.3.44 operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
    map< _Key, _Tp, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**5.908.3.45 operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Map list assignment operator.

**Parameters**

↩	An initializer_list.
↩	
↩	
↩	
↩	
↩	

This function fills a map with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned.

Definition at line 335 of file stl\_map.h.

**5.908.3.46 operator[]()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type& std::map< _Key, _Tp, _Compare, _Alloc >::operator[] (
    const key_type & __k ) [inline]
```

Subscript ( [] ) access to map data.

**Parameters**

↩ _k	The key for which data should be retrieved.
---------	---

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 490 of file `stl_map.h`.

**5.908.3.47 `rbegin()`** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 390 of file `stl_map.h`.

**5.908.3.48 `rbegin()`** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 399 of file `stl_map.h`.

**5.908.3.49 `rend()`** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 408 of file `stl_map.h`.

**5.908.3.50** `rend()` [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 417 of file `stl_map.h`.

**5.908.3.51** `size()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the size of the map.

Definition at line 468 of file `stl_map.h`.

**5.908.3.52** `swap()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::swap (
    map< _Key, _Tp, _Compare, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another map.

**Parameters**

<code>__x</code>	A map of the same element and allocator types.
------------------	--

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1121 of file `stl_map.h`.

**5.908.3.53** `upper_bound()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

**Returns**

Iterator pointing to the first element greater than key, or end().

Definition at line 1283 of file `stl_map.h`.

**5.908.3.54** `upper_bound()` [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

**Returns**

Iterator pointing to the first element greater than key, or end().

Definition at line 1289 of file `stl_map.h`.

**5.908.3.55** `upper_bound()` [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 1303 of file stl\_map.h.

#### 5.908.3.56 upper\_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 1309 of file stl\_map.h.

#### 5.908.3.57 value\_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::map< _Key, _Tp, _Compare, _Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 1149 of file stl\_map.h.

The documentation for this class was generated from the following file:



- [stl\\_map.h](#)

## 5.909 std::mask\_array< \_Tp > Class Template Reference

### Public Types

- typedef `_Tp` **value\_type**

### Public Member Functions

- [mask\\_array](#) (const [mask\\_array](#) &)
- void [operator%=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator%=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator&=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator&=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator\\*=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator\\*=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator+=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator+=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator-=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator-=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator/=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator/=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator<=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator<=](#) (const \_Expr< \_Dom, \_Tp > &) const
- [mask\\_array](#) & [operator=](#) (const [mask\\_array](#) &)
- void [operator=](#) (const [valarray](#)< \_Tp > &) const
- void [operator=](#) (const \_Tp &) const
- template<class \_Dom >  
void [operator=](#) (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Ex >  
void [operator=](#) (const \_Expr< \_Ex, \_Tp > &\_\_e) const
- void [operator>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator>=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator^=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator^=](#) (const \_Expr< \_Dom, \_Tp > &) const
- void [operator|=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void [operator|=](#) (const \_Expr< \_Dom, \_Tp > &) const

## Friends

- class **valarray**<\_Tp>

## 5.909.1 Detailed Description

```
template<class _Tp>
class std::mask_array<_Tp>
```

Reference to selected subset of an array.

A `mask_array` is a reference to the actual elements of an array specified by a bitmask in the form of an array of `bool`. The way to get a `mask_array` is to call `operator[]`(`valarray<bool>`) on a `valarray`. The returned `mask_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if a `mask_array` is obtained using the array `(false, true, false, true)` as an argument, the mask array has two elements referring to `array[1]` and `array[3]` in the underlying array.

## Parameters

<i>Tp</i>	Element type.
-----------	---------------

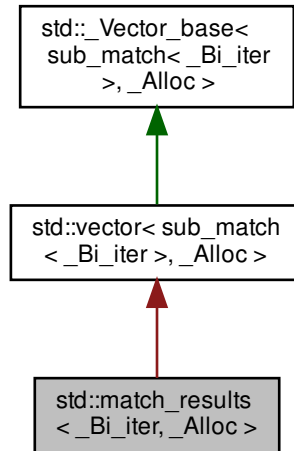
Definition at line 83 of file `valarray`.

The documentation for this class was generated from the following files:

- [valarray](#)
- [mask\\_array.h](#)

## 5.910 `std::match_results<_Bi_iter, _Alloc>` Class Template Reference

Inheritance diagram for `std::match_results<_Bi_iter, _Alloc>`:



### Public Member Functions

- `template<typename _Out_iter>`  
`_Out_iter format` (`_Out_iter __out`, `const match_results<_Bi_iter, _Alloc>::char_type *__fmt_first`, `const match_results<_Bi_iter, _Alloc>::char_type *__fmt_last`, `match_flag_type __flags`) `const`
- `bool ready` () `const`

### Private Types

- `typedef _Alloc_traits::const_pointer` `const_pointer`
- `typedef std::reverse_iterator< const_iterator >` `const_reverse_iterator`
- `typedef _Base::pointer` `pointer`
- `typedef std::reverse_iterator< iterator >` `reverse_iterator`

### Private Member Functions

- `pointer _M_allocate` (`size_t __n`)
- `pointer _M_allocate` (`size_t __n`)
- `pointer _M_allocate_and_copy` (`size_type __n`, `_ForwardIterator __first`, `_ForwardIterator __last`)
- `void _M_assign_aux` (`_InputIterator __first`, `_InputIterator __last`, `std::input_iterator_tag`)
- `void _M_assign_aux` (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)
- `void _M_assign_dispatch` (`_Integer __n`, `_Integer __val`, `__true_type`)
- `void _M_assign_dispatch` (`_InputIterator __first`, `_InputIterator __last`, `__false_type`)

- `size_type _M_check_len` (`size_type __n`, `const char *__s`) `const`
- `void _M_deallocate` (`pointer __p`, `size_t __n`)
- `void _M_deallocate` (`pointer __p`, `size_t __n`)
- `void _M_default_append` (`size_type __n`)
- `void _M_default_initialize` (`size_type __n`)
- `auto _M_emplace_aux` (`const_iterator __position`, `_Args &&... __args`) `-> iterator`
- `iterator _M_emplace_aux` (`const_iterator __position`, `_Args &&... __args`)
- `iterator _M_emplace_aux` (`const_iterator __position`, `value_type && __v`)
- `iterator _M_erase` (`iterator __position`)
- `iterator _M_erase` (`iterator __first`, `iterator __last`)
- `void _M_erase_at_end` (`pointer __pos`) `noexcept`
- `void _M_fill_assign` (`size_type __n`, `const value_type & __val`)
- `void _M_fill_initialize` (`size_type __n`, `const value_type & __value`)
- `void _M_fill_insert` (`iterator __pos`, `size_type __n`, `const value_type & __x`)
- `_Tp_alloc_type & _M_get_Tp_allocator` () `noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator` () `const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator` () `noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator` () `const noexcept`
- `void _M_initialize_dispatch` (`_Integer __n`, `_Integer __value`, `__true_type`)
- `void _M_initialize_dispatch` (`_InputIterator __first`, `_InputIterator __last`, `__false_type`)
- `void _M_insert_aux` (`iterator __position`, `_Arg && __arg`)
- `void _M_insert_dispatch` (`iterator __pos`, `_Integer __n`, `_Integer __val`, `__true_type`)
- `void _M_insert_dispatch` (`iterator __pos`, `_InputIterator __first`, `_InputIterator __last`, `__false_type`)
- `iterator _M_insert_rval` (`const_iterator __position`, `value_type && __v`)
- `void _M_range_check` (`size_type __n`) `const`
- `void _M_range_initialize` (`_InputIterator __first`, `_InputIterator __last`, `std::input_iterator_tag`)
- `void _M_range_initialize` (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)
- `void _M_range_insert` (`iterator __pos`, `_InputIterator __first`, `_InputIterator __last`, `std::input_iterator_tag`)
- `void _M_range_insert` (`iterator __pos`, `_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)
- `void _M_realloc_insert` (`iterator __position`, `_Args &&... __args`)
- `bool _M_shrink_to_fit` ()
- `void assign` (`size_type __n`, `const value_type & __val`)
- `void assign` (`_InputIterator __first`, `_InputIterator __last`)
- `void assign` (`initializer_list< value_type > __l`)
- `reference at` (`size_type __n`)
- `const_reference at` (`size_type __n`) `const`
- `reference back` () `noexcept`
- `const_reference back` () `const noexcept`
- `iterator begin` () `noexcept`
- `size_type capacity` () `const noexcept`
- `void clear` () `noexcept`
- `const_reverse_iterator cbegin` () `const noexcept`
- `const_reverse_iterator crend` () `const noexcept`
- `sub_match< _Bi_iter > * data` () `noexcept`
- `const sub_match< _Bi_iter > * data` () `const noexcept`
- `iterator emplace` (`const_iterator __position`, `_Args &&... __args`)
- `void emplace_back` (`_Args &&... __args`)
- `iterator end` () `noexcept`
- `iterator erase` (`const_iterator __position`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `reference front` () `noexcept`

- `const_reference front ()` const noexcept
- `allocator_type get_allocator ()` const noexcept
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `iterator insert (const_iterator __position, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const value_type &__x)`
- `iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `reference operator[] (size_type __n)` noexcept
- `const_reference operator[] (size_type __n)` const noexcept
- `void pop_back ()` noexcept
- `void push_back (const value_type &__x)`
- `void push_back (value_type &&__x)`
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rend ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- `void reserve (size_type __n)`
- `void resize (size_type __new_size)`
- `void resize (size_type __new_size, const value_type &__x)`
- `void shrink_to_fit ()`
- `void swap (vector &__x)` noexcept

#### Private Attributes

- `_Vector_impl _M_impl`

#### Friends

- `template<typename _Bp, typename _Ap, typename _Cp, typename _Rp, __detail::_RegexExecutorPolicy, bool >`  
`bool __detail::_regex_algo_impl (_Bp, _Bp, match_results< _Bp, _Ap > &, const basic_regex< _Cp, _Rp >`  
`&, regex_constants::match_flag_type)`
- `template<typename, typename, typename, bool >`  
`class __detail::_Executor`
- `template<typename, typename, typename >`  
`class regex_iterator`

#### 10.? Public Types

- `typedef sub_match< _Bi_iter > value_type`
- `typedef const value_type & const_reference`
- `typedef value_type & reference`
- `typedef _Base_type::const_iterator const_iterator`
- `typedef const_iterator iterator`
- `typedef __iter_traits::difference_type difference_type`
- `typedef allocator_traits< _Alloc >::size_type size_type`
- `typedef _Alloc allocator_type`
- `typedef __iter_traits::value_type char_type`
- `typedef std::basic_string< char_type > string_type`

## 28.10.1 Construction, Copying, and Destruction

- `match_results` (const `_Alloc` &\_\_a=`_Alloc`())
- `match_results` (const `match_results` &\_\_rhs)=default
- `match_results` (`match_results` &&\_\_rhs) noexcept=default
- `match_results` & `operator=` (const `match_results` &\_\_rhs)=default
- `match_results` & `operator=` (`match_results` &&\_\_rhs)=default
- `~match_results` ()

## 28.10.2 Size

- `size_type size` () const
- `size_type max_size` () const
- `bool empty` () const

## 10.3 Element Access

- `difference_type length` (size\_type \_\_sub=0) const
- `difference_type position` (size\_type \_\_sub=0) const
- `string_type str` (size\_type \_\_sub=0) const
- `const_reference operator[]` (size\_type \_\_sub) const
- `const_reference prefix` () const
- `const_reference suffix` () const
- `const_iterator begin` () const
- `const_iterator cbegin` () const
- `const_iterator end` () const
- `const_iterator cend` () const

## 10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter >`  
`_Out_iter format` (`_Out_iter` \_\_out, const `char_type` \*\_\_fmt\_first, const `char_type` \*\_\_fmt\_last, `match_flag_type` \_\_flags=`regex_constants::format_default`) const
- `template<typename _Out_iter, typename _St, typename _Sa >`  
`_Out_iter format` (`_Out_iter` \_\_out, const `basic_string`< `char_type`, `_St`, `_Sa` > &\_\_fmt, `match_flag_type` \_\_flags=`regex_constants::format_default`) const
- `template<typename _St, typename _Sa >`  
`basic_string`< `char_type`, `_St`, `_Sa` > `format` (const `basic_string`< `char_type`, `_St`, `_Sa` > &\_\_fmt, `match_flag_type` \_\_flags=`regex_constants::format_default`) const
- `string_type format` (const `char_type` \*\_\_fmt, `match_flag_type` \_\_flags=`regex_constants::format_default`) const

## 10.5 Allocator

- `allocator_type get_allocator` () const

## 10.6 Swap

- void `swap` (`match_results` &\_\_that)

### 5.910.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
class std::match_results< _Bi_iter, _Alloc >
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_↵match` member `matched` is always true. The `sub_match` object stored at index `n` denotes what matched the marked sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters [`first`, `second`) which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of the sequence that was searched.

Definition at line 39 of file `regex.h`.

### 5.910.2 Constructor & Destructor Documentation

#### 5.910.2.1 `match_results()` [1/3]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
    const _Alloc & __a = _Alloc() ) [inline], [explicit]
```

Constructs a default `match_results` container.

#### Postcondition

`size()` returns 0 and `str()` returns an empty string.

Definition at line 1605 of file `regex.h`.

### 5.910.2.2 match\_results() [2/3]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
    const match_results< _Bi_iter, _Alloc > & __rhs ) [default]
```

Copy constructs a match\_results.

### 5.910.2.3 match\_results() [3/3]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
    match_results< _Bi_iter, _Alloc > && __rhs ) [default], [noexcept]
```

Move constructs a match\_results.

### 5.910.2.4 ~match\_results()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::~~match_results ( ) [inline]
```

Destroys a match\_results object.

Definition at line 1634 of file regex.h.

## 5.910.3 Member Function Documentation

### 5.910.3.1 begin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::begin ( ) const [inline]
```

Gets an iterator to the start of the sub\_match collection.

Definition at line 1779 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::cbegin().



#### 5.910.3.2 cbegin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cbegin ( ) const [inline]
```

Gets an iterator to the start of the sub\_match collection.

Definition at line 1786 of file regex.h.

#### 5.910.3.3 cend()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cend ( ) const [inline]
```

Gets an iterator to one-past-the-end of the collection.

Definition at line 1800 of file regex.h.

#### 5.910.3.4 empty()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::empty ( ) const [inline]
```

Indicates if the match\_results contains no results.

##### Return values

<i>true</i>	The match_results object is empty.
<i>false</i>	The match_results object is not empty.

Definition at line 1675 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::end().

#### 5.910.3.5 end()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::end ( ) const [inline]
```

Gets an iterator to one-past-the-end of the collection.

Definition at line 1793 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::cend().

## 5.910.3.6 format() [1/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter >
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
    _Out_iter __out,
    const char_type * __fmt_first,
    const char_type * __fmt_last,
    match_flag_type __flags = regex_constants::format_default ) const
```

## Precondition

ready() == true

Referenced by std::match\_results< \_Bi\_iter >::format().

## 5.910.3.7 format() [2/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter , typename _St , typename _Sa >
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
    _Out_iter __out,
    const basic_string< char_type, _St, _Sa > & __fmt,
    match_flag_type __flags = regex_constants::format_default ) const [inline]
```

## Precondition

ready() == true

Definition at line 1829 of file regex.h.

## 5.910.3.8 format() [3/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _St , typename _Sa >
basic_string<char_type, _St, _Sa> std::match_results< _Bi_iter, _Alloc >::format (
    const basic_string< char_type, _St, _Sa > & __fmt,
    match_flag_type __flags = regex_constants::format_default ) const [inline]
```

## Precondition

ready() == true

Definition at line 1841 of file regex.h.

### 5.910.3.9 format() [ 4 / 4 ]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::format (
    const char_type * __fmt,
    match_flag_type __flags = regex_constants::format_default ) const [inline]
```

#### Precondition

ready() == true

Definition at line 1853 of file regex.h.

### 5.910.3.10 get\_allocator()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
allocator_type std::match_results< _Bi_iter, _Alloc >::get_allocator ( ) const [inline]
```

Gets a copy of the allocator.

Definition at line 1875 of file regex.h.

### 5.910.3.11 length()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::length (
    size_type __sub = 0 ) const [inline]
```

Gets the length of the indicated submatch.

#### Parameters

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

#### Precondition

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

Definition at line 1694 of file regex.h.

## 5.910.3.12 max\_size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::max_size ( ) const [inline]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark\_count() + 1 if a match was successful. Some matches may be empty.

## Returns

the number of matches found.

Definition at line 1666 of file regex.h.

## 5.910.3.13 operator=() [1/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results& std::match_results< _Bi_iter, _Alloc >::operator= (
    const match_results< _Bi_iter, _Alloc > & __rhs ) [default]
```

Assigns rhs to \*this.

## 5.910.3.14 operator=() [2/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results& std::match_results< _Bi_iter, _Alloc >::operator= (
    match_results< _Bi_iter, _Alloc > && __rhs ) [default]
```

Move-assigns rhs to \*this.

## 5.910.3.15 operator[]()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::operator[] (
    size_type __sub ) const [inline]
```

Gets a sub\_match reference for the match or submatch.

## Parameters

__sub	indicates the submatch.
-------	-------------------------

**Precondition**

`ready() == true`

This function gets a reference to the indicated submatch, or the entire match if `__sub` is zero.

If `__sub >= size()` then this function returns a `sub_match` with a special value indicating no submatch.

Definition at line 1737 of file `regex.h`.

**5.910.3.16 position()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::position (
    size_type __sub = 0 ) const [inline]
```

Gets the offset of the beginning of the indicated submatch.

**Parameters**

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

**Precondition**

`ready() == true`

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of `__sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Definition at line 1709 of file `regex.h`.

**5.910.3.17 prefix()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::prefix ( ) const [inline]
```

Gets a `sub_match` representing the match prefix.

**Precondition**

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1754 of file `regex.h`.

### 5.910.3.18 ready()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::ready ( ) const [inline]
```

Indicates if the match\_results is ready.

#### Return values

<i>true</i>	The object has a fully-established result state.
<i>false</i>	The object is not ready.

Definition at line 1645 of file regex.h.

### 5.910.3.19 size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::size ( ) const [inline]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark\_count() + 1 if a match was successful. Some matches may be empty.

#### Returns

the number of matches found.

Definition at line 1662 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::empty().

### 5.910.3.20 str()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::str (
    size_type __sub = 0 ) const [inline]
```

Gets the match or submatch converted to a string type.

#### Parameters

<i>__sub</i>	indicates the submatch.
--------------	-------------------------

**Precondition**

ready() == true

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

Definition at line 1722 of file `regex.h`.

**5.910.3.21 suffix()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::suffix ( ) const [inline]
```

Gets a `sub_match` representing the match suffix.

**Precondition**

ready() == true

This function gets a reference to a `sub_match` object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1769 of file `regex.h`.

**5.910.3.22 swap()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
void std::match_results< _Bi_iter, _Alloc >::swap (
    match_results< _Bi_iter, _Alloc > & __that ) [inline]
```

Swaps the contents of two `match_results`.

Definition at line 1889 of file `regex.h`.

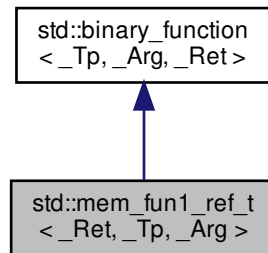
Referenced by `std::match_results< _Bi_iter >::swap()`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

## 5.911 std::mem\_fun1\_ref\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

## Public Member Functions

- **mem\_fun1\_ref\_t** (`_Ret`(`_Tp`::\* `__pf`)(`_Arg`))
- `_Ret` **operator()** (`_Tp` & `__r`, `_Arg` `__x`) const

## 5.911.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1287 of file `stl_function.h`.

## 5.911.2 Member Typedef Documentation



#### 5.911.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Arg , _Ret >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.911.2.2 result\_type

```
typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.911.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

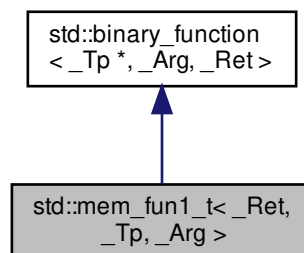
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.912 std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

Inheritance diagram for std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

## Public Member Functions

- **mem\_fun1\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg))
- **\_Ret operator()** (\_Tp \*\_\_p, \_Arg \_\_x) const

### 5.912.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1251 of file stl\_function.h.

### 5.912.2 Member Typedef Documentation

#### 5.912.2.1 first\_argument\_type

```
typedef _Tp * std::binary\_function< _Tp * , _Arg , _Ret >::first\_argument\_type [inherited]
```

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.912.2.2 result\_type

```
typedef _Ret std::binary\_function< _Tp * , _Arg , _Ret >::result\_type [inherited]
```

[result\\_type](#) is the return type

Definition at line 127 of file stl\_function.h.

### 5.912.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< _Tp * , _Arg , _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

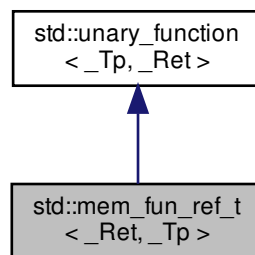
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.913 std::mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference

Inheritance diagram for std::mem\_fun\_ref\_t< \_Ret, \_Tp >:



#### Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

#### Public Member Functions

- **mem\_fun\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)())
- **\_Ret operator()** (\_Tp &\_\_r) const

#### 5.913.1 Detailed Description

```
template<typename _Ret, typename _Tp>  
class std::mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

Definition at line 1215 of file stl\_function.h.

## 5.913.2 Member Typedef Documentation

## 5.913.2.1 argument\_type

```
typedef _Tp std::unary_function< _Tp , _Ret >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 5.913.2.2 result\_type

```
typedef _Ret std::unary_function< _Tp , _Ret >::result_type [inherited]
```

result\_type is the return type

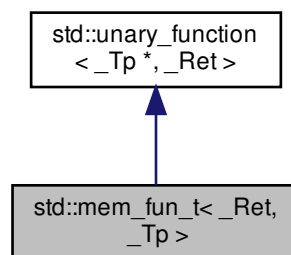
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.914 std::mem\_fun\_t&lt;\_Ret, \_Tp&gt; Class Template Reference

Inheritance diagram for std::mem\_fun\_t<\_Ret, \_Tp>:



## Public Types

- typedef \_Tp \* [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

## Public Member Functions

- **mem\_fun\_t** (\_Ret(\_Tp::\*\_\_pf)())
- **\_Ret operator()** (\_Tp \*\_\_p) const

### 5.914.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::mem_fun_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 1179 of file `stl_function.h`.

### 5.914.2 Member Typedef Documentation

#### 5.914.2.1 argument\_type

```
typedef _Tp * std::unary_function<_Tp * , _Ret >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.914.2.2 result\_type

```
typedef _Ret std::unary_function<_Tp * , _Ret >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.915 std::mersenne\_twister\_engine<\_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f> Class Template Reference

### Public Types

- typedef \_UIntType [result\\_type](#)

## Public Member Functions

- `mersenne_twister_engine` (`result_type` \_\_sd=default\_seed)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value>::type> mersenne_twister_engine` (`_Sseq` &\_\_q)
- `void discard` (unsigned long long \_\_z)
- `result_type operator()` ()
- `void seed` (`result_type` \_\_sd=default\_seed)
- `template<typename _Sseq> std::enable_if< std::is_class<_Sseq>::value>::type seed` (`_Sseq` &\_\_q)

## Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

## Static Public Attributes

- `static constexpr result_type default_seed`
- `static constexpr result_type initialization_multiplier`
- `static constexpr size_t mask_bits`
- `static constexpr size_t shift_size`
- `static constexpr size_t state_size`
- `static constexpr result_type tempering_b`
- `static constexpr result_type tempering_c`
- `static constexpr result_type tempering_d`
- `static constexpr size_t tempering_l`
- `static constexpr size_t tempering_s`
- `static constexpr size_t tempering_t`
- `static constexpr size_t tempering_u`
- `static constexpr size_t word_size`
- `static constexpr result_type xor_mask`

## Friends

- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &__os, const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> &__x)`
- `bool operator==` (const `mersenne_twister_engine` &\_\_lhs, const `mersenne_twister_engine` &\_\_rhs)
- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> &__x)`

### 5.915.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
        _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined mt19937 class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

#### Template Parameters

<code>__w</code>	Word size, the number of bits in each element of the state vector.
<code>__n</code>	The degree of recursion.
<code>__m</code>	The period parameter.
<code>__r</code>	The separation point bit index.
<code>__a</code>	The last row of the twist matrix.
<code>__u</code>	The first right-shift tempering matrix parameter.
<code>__d</code>	The first right-shift tempering matrix mask.
<code>__s</code>	The first left-shift tempering matrix parameter.
<code>__b</code>	The first left-shift tempering matrix mask.
<code>__t</code>	The second left-shift tempering matrix parameter.
<code>__c</code>	The second left-shift tempering matrix mask.
<code>__l</code>	The second right-shift tempering matrix parameter.
<code>__f</code>	Initialization multiplier.

Definition at line 437 of file random.h.

### 5.915.2 Member Typedef Documentation

### 5.915.2.1 result\_type

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type
```

The type of the generated random value.

Definition at line 440 of file random.h.

## 5.915.3 Constructor & Destructor Documentation

### 5.915.3.1 mersenne\_twister\_engine()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value> ::type>
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine (
    _Sseq & __q ) [inline], [explicit]
```

Constructs a mersenne\_twister\_engine random number generator engine seeded from the seed sequence \_\_q.

#### Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

Definition at line 501 of file random.h.

## 5.915.4 Member Function Documentation

### 5.915.4.1 discard()

```
template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```



```
void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, ↵
__c, __l, __f >::discard (
    unsigned long long __z )
```

Discard a sequence of random numbers.

Definition at line 432 of file bits/random.tcc.

#### 5.915.4.2 max()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t ↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType ↵
__f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, ↵
__u, __d, __s, __b, __t, __c, __l, __f >::max ( ) [inline], [static]
```

Gets the largest possible value in the output range.

Definition at line 522 of file random.h.

#### 5.915.4.3 min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t ↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType ↵
__f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, ↵
__u, __d, __s, __b, __t, __c, __l, __f >::min ( ) [inline], [static]
```

Gets the smallest possible value in the output range.

Definition at line 515 of file random.h.

### 5.915.5 Friends And Related Function Documentation

#### 5.915.5.1 operator<<

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t ↵
__t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType ↵
__f>
template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 ↵
__a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, ↵
size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, ↵
__d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x ) [friend]
```

Inserts the current state of a % mersenne\_twister\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A % mersenne_twister_engine random number generator engine.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### 5.915.5.2 operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool operator== (
    const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs,
    const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs ) [friend]
```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for equality.

#### Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 547 of file random.h.

#### 5.915.5.3 operator>>

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > &__is,
    std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x ) [friend]
```

Extracts the current state of a % mersenne\_twister\_engine random number generator engine `__x` from the input stream `__is`.

## Parameters

<code>_is</code>	An input stream.
<code>_x</code>	A % mersenne_twister_engine random number generator engine.

## Returns

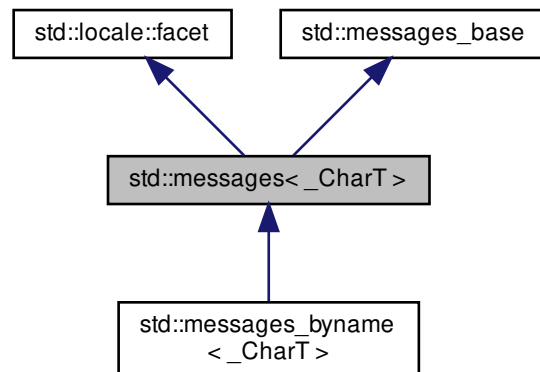
The input stream with the state of `_x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.916 `std::messages<_CharT>` Class Template Reference

Inheritance diagram for `std::messages<_CharT>`:



## Public Types

- typedef int **catalog**
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

## Public Member Functions

- [messages](#) (size\_t \_\_refs=0)
- [messages](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- [string\\_type](#) **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const [string\\_type](#) &\_\_s) const
- catalog **open** (const [basic\\_string](#)< char > &\_\_s, const [locale](#) &\_\_loc) const
- catalog **open** (const [basic\\_string](#)< char > &, const [locale](#) &, const char \*) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~messages](#) ()
- [string\\_type](#) **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const [string\\_type](#) &\_\_msg) const
- template<>  
void **do\_close** (catalog) const
- template<>  
void **do\_close** (catalog) const
- virtual void **do\_close** (catalog) const
- virtual [string\\_type](#) **do\_get** (catalog, int, int, const [string\\_type](#) &\_\_dfault) const
- template<>  
[string](#) **do\_get** (catalog, int, int, const [string](#) &) const
- template<>  
[wstring](#) **do\_get** (catalog, int, int, const [wstring](#) &) const
- template<>  
[messages](#)< char >::catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const
- template<>  
[messages](#)< wchar\_t >::catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const
- virtual catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_messages**
- const char \* **\_M\_name\_messages**

### 5.916.1 Detailed Description

```
template<typename _CharT>
class std::messages< _CharT >
```

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar\_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1799 of file locale\_facets\_nonio.h.

### 5.916.2 Member Typedef Documentation

#### 5.916.2.1 char\_type

```
template<typename _CharT >
typedef _CharT std::messages< _CharT >::char_type
```

Public typedefs.

Definition at line 1805 of file locale\_facets\_nonio.h.

#### 5.916.2.2 string\_type

```
template<typename _CharT >
typedef basic_string<_CharT> std::messages< _CharT >::string_type
```

Public typedefs.

Definition at line 1806 of file locale\_facets\_nonio.h.

### 5.916.3 Constructor & Destructor Documentation

#### 5.916.3.1 messages() [1/2]

```
template<typename _CharT >
std::messages< _CharT >::messages (
    size_t __refs = 0 ) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 44 of file messages\_members.h.

## 5.916.3.2 messages() [2/2]

```
template<typename _CharT >
std::messages<_CharT>::messages (
    __c_locale __cloc,
    const char * __s,
    size_t __refs = 0 ) [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

<code>__cloc</code>	The C locale.
<code>__s</code>	The name of a locale.
<code>__refs</code>	RefCount to pass to the base class.

Definition at line 50 of file messages\_members.h.

## 5.916.3.3 ~messages()

```
template<typename _CharT >
std::messages<_CharT>::~~messages ( ) [protected], [virtual]
```

Destructor.

Definition at line 79 of file messages\_members.h.

## 5.916.4 Member Function Documentation

#### 5.916.4.1 do\_get()

```
template<>
string std::messages< char >::do_get (
    catalog ,
    int ,
    int ,
    const string & ) const [protected]
```

Specializations for required instantiations.

#### 5.916.5 Member Data Documentation

##### 5.916.5.1 id

```
template<typename _CharT >
locale::id std::messages< _CharT >::id [static]
```

Numpunct facet id.

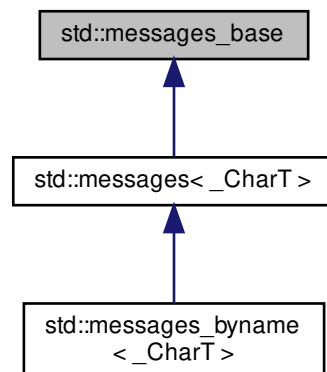
Definition at line 1817 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

#### 5.917 std::messages\_base Struct Reference

Inheritance diagram for std::messages\_base:



## Public Types

- typedef int **catalog**

## 5.917.1 Detailed Description

Messages facet base class providing catalog typedef.

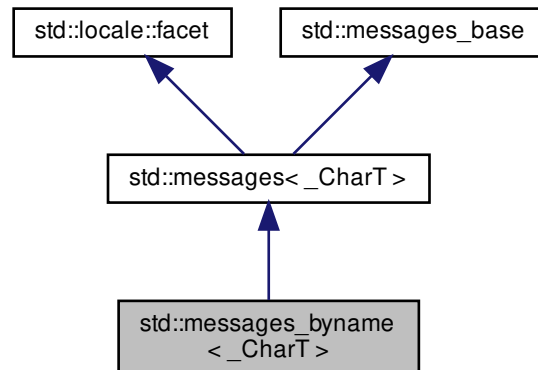
Definition at line 1770 of file locale\_facets\_nonio.h.

The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.918 std::messages\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::messages\_byname<\_CharT>:



## Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef [basic\\_string](#)<`_CharT`> **string\_type**



### Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **messages\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- [string\\_type](#) **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const [string\\_type](#) &\_\_s) const
- catalog **open** (const [basic\\_string](#)< char > &\_\_s, const [locale](#) &\_\_loc) const
- catalog **open** (const [basic\\_string](#)< char > &, const [locale](#) &, const char \*) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- [string\\_type](#) **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const [string\\_type](#) &\_\_msg) const
- template<>  
void **do\_close** (catalog) const
- template<>  
void **do\_close** (catalog) const
- virtual void **do\_close** (catalog) const
- virtual [string\\_type](#) **do\_get** (catalog, int, int, const [string\\_type](#) &\_\_dfault) const
- template<>  
[string](#) **do\_get** (catalog, int, int, const [string](#) &) const
- template<>  
[wstring](#) **do\_get** (catalog, int, int, const [wstring](#) &) const
- template<>  
[messages](#)< char >::catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const
- template<>  
[messages](#)< wchar\_t >::catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const
- virtual catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_messages**
- const char \* **\_M\_name\_messages**

### 5.918.1 Detailed Description

```
template<typename _CharT>
class std::messages_byname<_CharT>
```

class messages\_byname [22.2.7.2].

Definition at line 1983 of file locale\_facets\_nonio.h.

### 5.918.2 Member Function Documentation

#### 5.918.2.1 do\_get()

```
template<>
string std::messages< char >::do_get (
    catalog ,
    int ,
    int ,
    const string & ) const [protected], [inherited]
```

Specializations for required instantiations.

### 5.918.3 Member Data Documentation

#### 5.918.3.1 id

```
template<typename _CharT>
locale::id std::messages<_CharT>::id [static], [inherited]
```

Numpunct facet id.

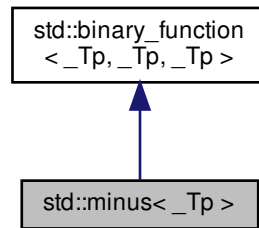
Definition at line 1817 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

### 5.919 std::minus<\_Tp> Struct Template Reference

Inheritance diagram for std::minus<\_Tp>:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR _Tp operator()(const _Tp &__x, const _Tp &__y) const`

#### 5.919.1 Detailed Description

```
template<typename _Tp>  
struct std::minus<_Tp>
```

One of the [math functors](#).

Definition at line 150 of file `stl_function.h`.

#### 5.919.2 Member Typedef Documentation

5.919.2.1 `first_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

5.919.2.2 `result_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

5.919.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

5.920 `std::minus< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
`_GLIBCXX14_CONSTEXPR` auto **operator()** (`_Tp` && `_t`, `_Up` && `_u`) const noexcept(noexcept(`std::forward`<  
`_Tp`>(`_t`) - `std::forward`< `_Up`>(`_u`))) -> decltype(`std::forward`< `_Tp`>(`_t`) - `std::forward`< `_Up`>(`_u`))

### 5.920.1 Detailed Description

```
template<>
struct std::minus< void >
```

One of the [math functors](#).

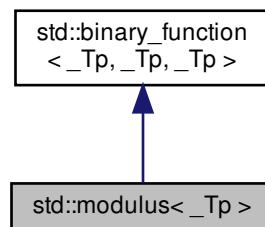
Definition at line 245 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.921 std::modulus< \_Tp > Struct Template Reference

Inheritance diagram for `std::modulus< _Tp >`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR _Tp operator() (const _Tp &__x, const _Tp &__y) const`

### 5.921.1 Detailed Description

```
template<typename _Tp>
struct std::modulus< _Tp >
```

One of the [math functors](#).

Definition at line 159 of file `stl_function.h`.

## 5.921.2 Member Typedef Documentation

## 5.921.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

## 5.921.2.2 result\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 5.921.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.922 std::modulus&lt; void &gt; Struct Template Reference

## Public Types

- typedef \_\_is\_transparent **is\_transparent**

## Public Member Functions

- template<typename \_Tp, typename \_Up >  
\_GLIBCXX14\_CONSTEXPR auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward<\_Tp>(\_\_t) % std::forward<\_Up>(\_\_u))) -> decltype(std::forward<\_Tp>(\_\_t) % std::forward<\_Up>(\_\_u))

### 5.922.1 Detailed Description

```
template<>
struct std::modulus< void >
```

One of the [math functors](#).

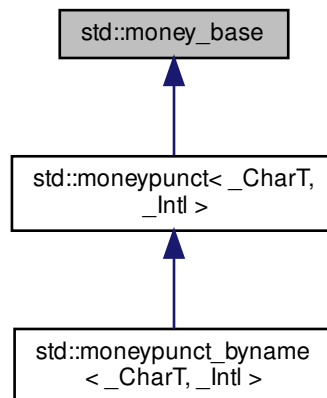
Definition at line 290 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.923 std::money\_base Class Reference

Inheritance diagram for `std::money_base`:



#### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- enum **part** { **none**, **space**, **symbol**, **sign**, **value** }

#### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

#### Static Public Attributes

- static const char \* `_S_atoms`
- static const pattern `_S_default_pattern`

#### 5.923.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. `symbol`, `sign`, and `value` must be present and the remaining field must contain either none or space.

#### See also

`moneypunct::pos_format()` and `moneypunct::neg_format()` for details of how these fields are interpreted.

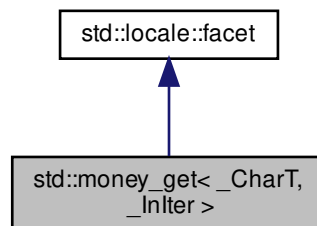
Definition at line 928 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.924 `std::money_get<_CharT, _InIter >` Class Template Reference

Inheritance diagram for `std::money_get<_CharT, _InIter >`:



#### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)
- typedef [basic\\_string<\\_CharT >](#) [string\\_type](#)



## Public Member Functions

- `money_get` (size\_t \_\_refs=0)
- `template<bool _Intl>`  
`_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11 _Intl _M_extract` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string & \_\_units) const
- `iter_type get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long double & \_\_units) const
- `iter_type get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string\_type & \_\_digits) const

## Static Public Attributes

- static `locale::id` id

## Protected Member Functions

- virtual `~money_get` ()
- `template<bool _Intl>`  
`iter_type _M_extract` (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string & \_\_digits) const
- virtual `iter_type do_get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long double & \_\_units) const
- virtual `iter_type do_get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string\_type & \_\_digits) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale & \_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale & \_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### 5.924.1 Detailed Description

```
template<typename _CharT, typename _Intl>
class std::money_get< _CharT, _Intl >
```

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

The `money_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_get` facet.

Definition at line 1468 of file `locale_facets_nonio.h`.

### 5.924.2 Member Typedef Documentation

#### 5.924.2.1 char\_type

```
template<typename _CharT , typename _InIter >
typedef _CharT std::money_get< _CharT, _InIter >::char_type
```

Public typedefs.

Definition at line 1474 of file locale\_facets\_nonio.h.

#### 5.924.2.2 iter\_type

```
template<typename _CharT , typename _InIter >
typedef _InIter std::money_get< _CharT, _InIter >::iter_type
```

Public typedefs.

Definition at line 1475 of file locale\_facets\_nonio.h.

#### 5.924.2.3 string\_type

```
template<typename _CharT , typename _InIter >
typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type
```

Public typedefs.

Definition at line 1476 of file locale\_facets\_nonio.h.

### 5.924.3 Constructor & Destructor Documentation

#### 5.924.3.1 money\_get()

```
template<typename _CharT , typename _InIter >
std::money_get< _CharT, _InIter >::money_get (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1490 of file `locale_facets_nonio.h`.

#### 5.924.3.2 `~money_get()`

```
template<typename _CharT , typename _InIter >
virtual std::money_get< _CharT, _InIter >::~~money_get ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 1558 of file `locale_facets_nonio.h`.

#### 5.924.4 Member Function Documentation

##### 5.924.4.1 `do_get()` [1/2]

```
template<typename _CharT , typename _InIter >
_InIter std::money_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __units ) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

#### See also

`get()` for details.

Definition at line 371 of file `locale_facets_nonio.tcc`.

References `std::basic_string`< `_CharT`, `_Traits`, `_Alloc` >::`c_str`().

Referenced by `std::money_get`< `_CharT`, `_InIter` >::`get`().

## 5.924.4.2 do\_get() [2/2]

```
template<typename _CharT , typename _InIter >
_InIter std::money_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    string_type & __digits ) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

## See also

get() for details.

Definition at line 384 of file locale\_facets\_nonio.tcc.

References `std::ios_base::M_getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

## 5.924.4.3 get() [1/2]

```
template<typename _CharT , typename _InIter >
iter_type std::money_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __units ) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `units` as an integral value `moneypunct::frac_digits()` \* the actual amount. For example, the string \$10.01 in a US locale would store 1001 in `units`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. `units` is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

## Parameters

<code>__s</code>	Start of characters to parse.
<code>__end</code>	End of characters to parse.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__err</code>	Error field to set if parsing fails.
<code>__units</code>	Place to store result of parsing.

## Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1520 of file `locale_facets_nonio.h`.

References `std::money_get<_CharT, _InIter >::do_get()`.

5.924.4.4 `get()` [2/2]

```
template<typename _CharT , typename _InIter >
iter_type std::money_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    bool __intl,
    ios_base & __io,
    ios_base::iostate & __err,
    string_type & __digits ) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

## Parameters

<code>__s</code>	Start of characters to parse.
<code>__end</code>	End of characters to parse.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__err</code>	Error field to set if parsing fails.
<code>__digits</code>	Place to store result of parsing.

### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1551 of file locale\_facets\_nonio.h.

References `std::money_get< _CharT, _InIter >::do_get()`.

## 5.924.5 Member Data Documentation

### 5.924.5.1 id

```
template<typename _CharT , typename _InIter >
locale::id std::money_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

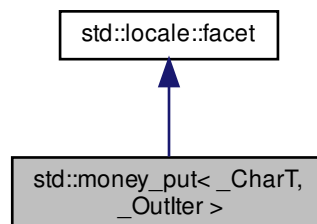
Definition at line 1480 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.925 std::money\_put< \_CharT, \_OutIter > Class Template Reference

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_Outlter` `iter_type`
- typedef `basic_string<_CharT>` `string_type`

## Public Member Functions

- `money_put` (`size_t __refs=0`)
- `template<bool __Intl>`  
`_Outlter _M_insert` (`iter_type __s`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`
- `iter_type put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `long double __units`) `const`
- `iter_type put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_put` ()
- `template<bool __Intl>`  
`iter_type _M_insert` (`iter_type __s`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`
- virtual `iter_type do_put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `long double __units`) `const`
- virtual `iter_type do_put` (`iter_type __s`, `bool __intl`, `ios_base & __io`, `char_type __fill`, `const string_type & __digits`) `const`

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (`_c_locale & __cloc`) `throw ()`
- static void `_S_create_c_locale` (`_c_locale & __cloc`, `const char * __s`, `_c_locale __old=0`)
- static void `_S_destroy_c_locale` (`_c_locale & __cloc`)
- static `_c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () `throw ()`
- static `_c_locale _S_lc_ctype_c_locale` (`_c_locale __cloc`, `const char * __s`)

## 5.925.1 Detailed Description

```
template<typename _CharT, typename _Outlter>
class std::money_put<_CharT, _Outlter>
```

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount.

The `money_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_put` facet.

Definition at line 1621 of file `locale_facets_nonio.h`.

## 5.925.2 Member Typedef Documentation

### 5.925.2.1 char\_type

```
template<typename _CharT , typename _OutIter >
typedef _CharT std::money_put< _CharT, _OutIter >::char_type
```

Public typedefs.

Definition at line 1626 of file locale\_facets\_nonio.h.

### 5.925.2.2 iter\_type

```
template<typename _CharT , typename _OutIter >
typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

Definition at line 1627 of file locale\_facets\_nonio.h.

### 5.925.2.3 string\_type

```
template<typename _CharT , typename _OutIter >
typedef basic_string<_CharT> std::money_put< _CharT, _OutIter >::string_type
```

Public typedefs.

Definition at line 1628 of file locale\_facets\_nonio.h.

## 5.925.3 Constructor & Destructor Documentation

### 5.925.3.1 money\_put()

```
template<typename _CharT , typename _OutIter >
std::money_put< _CharT, _OutIter >::money_put (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.



**Parameters**

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1642 of file `locale_facets_nonio.h`.

**5.925.3.2 ~money\_put()**

```
template<typename _CharT , typename _OutIter >
virtual std::money_put< _CharT, _OutIter >::~~money_put ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 1692 of file `locale_facets_nonio.h`.

**5.925.4 Member Function Documentation****5.925.4.1 do\_put() [1/2]**

```
template<typename _CharT , typename _OutIter >
_OutIter std::money_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    long double __units ) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write `$10.01` to `__s`.

This function is a hook for derived classes to change the value returned.

**See also**

`put()`.

**Parameters**

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>units</code>	Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 577 of file locale\_facets\_nonio.tcc.

References std::ios\_base::getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

Referenced by std::money\_put< \_CharT, \_OutIter >::put().

**5.925.4.2 do\_put()** [2/2]

```
template<typename _CharT , typename _OutIter >
_OutIter std::money_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    const string_type & __digits ) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the string 1001 in a US locale would write \$10.01 to \_\_s.

This function is a hook for derived classes to change the value returned.

**See also**

put().

**Parameters**

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use_facet<moneypunct<CharT,intl> >.
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	char_type to use for padding.
<code>__digits</code>	Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 615 of file locale\_facets\_nonio.tcc.

5.925.4.3 `put()` [1/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::money_put< _CharT, _OutIter >::put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    long double __units ) const [inline]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write \$10.01 to `__s`.

This function works by returning the result of `do_put()`.

## Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__units</code>	Place to store result of parsing.

## Returns

Iterator after writing.

Definition at line 1662 of file `locale_facets_nonio.h`.

References `std::money_put< _CharT, _OutIter >::do_put()`.

5.925.4.4 `put()` [2/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::money_put< _CharT, _OutIter >::put (
    iter_type __s,
    bool __intl,
    ios_base & __io,
    char_type __fill,
    const string_type & __digits ) const [inline]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string 1001 in a US locale would write \$10.01 to `__s`.

This function works by returning the result of `do_put()`.

## Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet&lt;money_punct&lt;CharT,intl&gt;&gt;</code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__digits</code>	Place to store result of parsing.

## Returns

Iterator after writing.

Definition at line 1685 of file `locale_facets_nonio.h`.

References `std::money_put< _CharT, _OutIter >::do_put()`.

## 5.925.5 Member Data Documentation

## 5.925.5.1 id

```
template<typename _CharT , typename _OutIter >
locale::id std::money_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

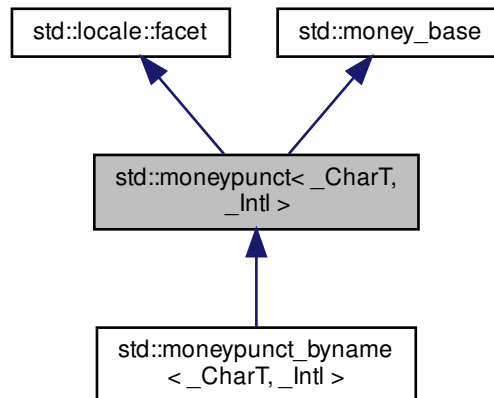
Definition at line 1632 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.926 std::moneypunct< \_CharT, \_Intl > Class Template Reference

Inheritance diagram for std::moneypunct< \_CharT, \_Intl >:



### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache< \_CharT, \_Intl > **\_\_cache\_type**
- enum **part** {  
**none**, **space**, **symbol**, **sign**,  
**value** }
- typedef \_CharT **char\_type**
- typedef **basic\_string**< \_CharT > **string\_type**

### Public Member Functions

- **moneypunct** (size\_t \_\_refs=0)
- **moneypunct** (\_\_cache\_type \*\_\_cache, size\_t \_\_refs=0)
- **moneypunct** (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- **string\_type** **curr\_symbol** () const
- **char\_type** **decimal\_point** () const
- int **frac\_digits** () const
- **string** **grouping** () const
- **string\_type** **negative\_sign** () const
- **string\_type** **positive\_sign** () const
- **char\_type** **thousands\_sep** () const
- pattern **pos\_format** () const
- pattern **neg\_format** () const

## Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

## Protected Member Functions

- virtual [~moneypunct](#) ()
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type](#) **do\_curr\_symbol** () const
- virtual [char\\_type](#) **do\_decimal\_point** () const
- virtual int **do\_frac\_digits** () const
- virtual [string](#) **do\_grouping** () const
- virtual pattern **do\_neg\_format** () const
- virtual [string\\_type](#) **do\_negative\_sign** () const
- virtual pattern **do\_pos\_format** () const
- virtual [string\\_type](#) **do\_positive\_sign** () const
- virtual [char\\_type](#) **do\_thousands\_sep** () const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.926.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct<_CharT, _Intl>
```

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 1024 of file locale\_facets\_nonio.h.

## 5.926.2 Member Typedef Documentation

### 5.926.2.1 char\_type

```
template<typename _CharT, bool _Intl>
typedef _CharT std::moneypunct< _CharT, _Intl >::char_type
```

Public typedefs.

Definition at line 1030 of file locale\_facets\_nonio.h.

### 5.926.2.2 string\_type

```
template<typename _CharT, bool _Intl>
typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type
```

Public typedefs.

Definition at line 1031 of file locale\_facets\_nonio.h.

## 5.926.3 Constructor & Destructor Documentation

### 5.926.3.1 moneypunct() [1/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1053 of file locale\_facets\_nonio.h.

### 5.926.3.2 moneypunct() [2/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct<_CharT, _Intl>::moneypunct (
    __cache_type * __cache,
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is an internal constructor.

#### Parameters

<code>__cache</code>	Cache for optimization.
<code>__refs</code>	Passed to the base facet class.

Definition at line 1066 of file locale\_facets\_nonio.h.

### 5.926.3.3 moneypunct() [3/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct<_CharT, _Intl>::moneypunct (
    __c_locale __cloc,
    const char * __s,
    size_t __refs = 0 ) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

<code>__cloc</code>	The C locale.
<code>__s</code>	The name of a locale.
<code>__refs</code>	Passed to the base facet class.

Definition at line 1081 of file locale\_facets\_nonio.h.

### 5.926.3.4 ~moneypunct()

```
template<typename _CharT, bool _Intl>
virtual std::moneypunct<_CharT, _Intl>::~~moneypunct ( ) [protected], [virtual]
```

Destructor.



#### 5.926.4 Member Function Documentation

##### 5.926.4.1 curr\_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol ( ) const [inline]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `returning moneypunct<char_↵_type>::do_curr_symbol()`.

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1151 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

##### 5.926.4.2 decimal\_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point ( ) const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning moneypunct<char_↵type>::do_decimal_point()`.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

#### 5.926.4.3 do\_curr\_symbol()

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_curr_symbol ( ) const [inline], [protected],
[virtual]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

#### See also

`curr_symbol()` for details.

#### Returns

*string\_type* representing a currency symbol.

Definition at line 1297 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::curr_symbol()`.

#### 5.926.4.4 do\_decimal\_point()

```
template<typename _CharT, bool _Intl>
virtual char\_type std::moneypunct< _CharT, _Intl >::do_decimal_point ( ) const [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1259 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::decimal_point()`.

#### 5.926.4.5 do\_frac\_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits ( ) const [inline], [protected],
[virtual]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

#### See also

frac\_digits() for details.

#### Returns

Number of digits in amount fraction.

Definition at line 1337 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::frac\_digits().

#### 5.926.4.6 do\_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const [inline], [protected],
[virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

#### See also

grouping() for details.

#### Returns

String representing grouping specification.

Definition at line 1284 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

#### 5.926.4.7 do\_neg\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

##### See also

neg\_format() for details.

##### Returns

Pattern for money values.

Definition at line 1365 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::neg\_format().

#### 5.926.4.8 do\_negative\_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const [inline], [protected],
[virtual]
```

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

##### See also

negative\_sign() for details.

##### Returns

*string\_type* representing a negative sign.

Definition at line 1323 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::negative\_sign().

#### 5.926.4.9 do\_pos\_format()

```
template<typename _CharT, bool _Intl>  
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const [inline], [protected],  
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

#### See also

pos\_format() for details.

#### Returns

Pattern for money values.

Definition at line 1351 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::pos\_format().

#### 5.926.4.10 do\_positive\_sign()

```
template<typename _CharT, bool _Intl>  
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const [inline], [protected],  
[virtual]
```

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

#### See also

positive\_sign() for details.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1310 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::positive\_sign().

## 5.926.4.11 do\_thousands\_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep ( ) const [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1271 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::thousands\_sep().

## 5.926.4.12 frac\_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct<_CharT, _Intl>::frac_digits ( ) const [inline]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char\_type>::do\_frac\_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac\_digits() digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1201 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl>::do\_frac\_digits().

#### 5.926.4.13 grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping ( ) const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

##### Returns

string representing grouping specification.

Definition at line 1138 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_grouping()`.

#### 5.926.4.14 neg\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format ( ) const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

##### Returns

Pattern for money values.

Definition at line 1241 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_neg_format()`.

## 5.926.4.15 negative\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct<_CharT, _Intl>::negative_sign ( ) const [inline]
```

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char\_type>::do\_negative\_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg\_format() and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1185 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl>::do\_negative\_sign().

## 5.926.4.16 pos\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::pos_format ( ) const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

**Returns**

Pattern for money values.

Definition at line 1237 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl>::do\_pos\_format().



#### 5.926.4.17 positive\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign ( ) const [inline]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `returning moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

##### Returns

*string\_type* representing a positive sign.

Definition at line 1168 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_positive_sign()`.

#### 5.926.4.18 thousands\_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const [inline]
```

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `returning moneypunct<char_type>::do_thousands_sep()`.

##### Returns

`char_type` representing a thousands separator.

Definition at line 1108 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_thousands_sep()`.

#### 5.926.5 Member Data Documentation

## 5.926.5.1 id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id [static]
```

Numpunct facet id.

Definition at line 1043 of file locale\_facets\_nonio.h.

## 5.926.5.2 intl

```
template<typename _CharT, bool _Intl>
const bool std::moneypunct< _CharT, _Intl >::intl [static]
```

This value is provided by the standard, but no reason for its existence.

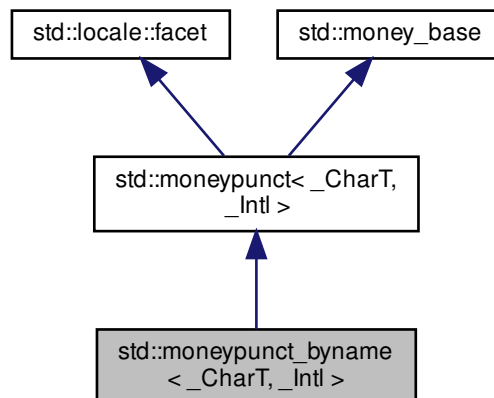
Definition at line 1041 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.927 std::moneypunct\_byname&lt; \_CharT, \_Intl &gt; Class Template Reference

Inheritance diagram for std::moneypunct\_byname< \_CharT, \_Intl >:



### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache< \_CharT, \_Intl > **\_\_cache\_type**
- typedef \_CharT **char\_type**
- enum **part** {  
    **none**, **space**, **symbol**, **sign**,  
    **value** }
- typedef [basic\\_string](#)< \_CharT > **string\_type**

### Public Member Functions

- **moneypunct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
  - **moneypunct\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
  - [string\\_type](#) **curr\_symbol** () const
  - [char\\_type](#) **decimal\_point** () const
  - int **frac\_digits** () const
  - [string](#) **grouping** () const
  - [string\\_type](#) **negative\_sign** () const
  - [string\\_type](#) **positive\_sign** () const
  - [char\\_type](#) **thousands\_sep** () const
- 
- pattern [pos\\_format](#) () const
  - pattern [neg\\_format](#) () const

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

## Protected Member Functions

- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type do\\_curr\\_symbol](#) () const
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual int [do\\_frac\\_digits](#) () const
- virtual [string do\\_grouping](#) () const
- virtual pattern [do\\_neg\\_format](#) () const
- virtual [string\\_type do\\_negative\\_sign](#) () const
- virtual pattern [do\\_pos\\_format](#) () const
- virtual [string\\_type do\\_positive\\_sign](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.927.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct_byname<_CharT, _Intl>
```

class moneypunct\_byname [22.2.6.4].

Definition at line 1414 of file locale\_facets\_nonio.h.

## 5.927.2 Member Function Documentation

#### 5.927.2.1 curr\_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol ( ) const [inline], [inherited]
```

Return currency symbol string.

This function returns a string\_type to use as a currency symbol. It does so by returning returning moneypunct<char↵\_type>::do\_curr\_symbol().

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1151 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_curr\_symbol().

#### 5.927.2.2 decimal\_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point ( ) const [inline], [inherited]
```

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning moneypunct<char↵\_type>::do\_decimal\_point().

##### Returns

*char\_type* representing a decimal point.

Definition at line 1095 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_decimal\_point().

#### 5.927.2.3 do\_curr\_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol ( ) const [inline], [protected],
[virtual], [inherited]
```

Return currency symbol string.

This function returns a string\_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

##### See also

curr\_symbol() for details.

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1297 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::curr\_symbol().

#### 5.927.2.4 do\_decimal\_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct<_CharT, _Intl >::do_decimal_point ( ) const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1259 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl >::decimal_point()`.

#### 5.927.2.5 do\_frac\_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct<_CharT, _Intl >::do_frac_digits ( ) const [inline], [protected],
[virtual], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

##### See also

`frac_digits()` for details.

##### Returns

Number of digits in amount fraction.

Definition at line 1337 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl >::frac_digits()`.

#### 5.927.2.6 do\_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const [inline], [protected],
[virtual], [inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1284 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

#### 5.927.2.7 do\_neg\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

##### See also

neg\_format() for details.

##### Returns

Pattern for money values.

Definition at line 1365 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::neg\_format().

### 5.927.2.8 do\_negative\_sign()

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const [inline], [protected],
[virtual], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

#### See also

`negative_sign()` for details.

#### Returns

*string\_type* representing a negative sign.

Definition at line 1323 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::negative_sign()`.

### 5.927.2.9 do\_pos\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

#### See also

`pos_format()` for details.

#### Returns

Pattern for money values.

Definition at line 1351 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.



#### 5.927.2.10 do\_positive\_sign()

```
template<typename _CharT, bool _Intl>
virtual string\_type std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const [inline], [protected],
[virtual], [inherited]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

#### See also

`positive_sign()` for details.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1310 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::positive_sign()`.

#### 5.927.2.11 do\_thousands\_sep()

```
template<typename _CharT, bool _Intl>
virtual char\_type std::moneypunct< _CharT, _Intl >::do_thousands_sep ( ) const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a thousands separator.

Definition at line 1271 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::thousands_sep()`.

#### 5.927.2.12 frac\_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct<_CharT, _Intl >::frac_digits ( ) const [inline], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `returning moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

##### Returns

Number of digits in amount fraction.

Definition at line 1201 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

#### 5.927.2.13 grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct<_CharT, _Intl >::grouping ( ) const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

##### Returns

string representing grouping specification.

Definition at line 1138 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_grouping()`.

#### 5.927.2.14 `neg_format()`

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format ( ) const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `std::moneypunct<char_type>::do_pos_format()` or `std::moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

##### Returns

Pattern for money values.

Definition at line 1241 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_neg_format()`.

#### 5.927.2.15 `negative_sign()`

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign ( ) const [inline], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `std::moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

##### Returns

*string\_type* representing a negative sign.

Definition at line 1185 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_negative_sign()`.

### 5.927.2.16 pos\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format ( ) const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

#### Returns

Pattern for money values.

Definition at line 1237 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_pos\_format().

### 5.927.2.17 positive\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign ( ) const [inline], [inherited]
```

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char\_type>::do\_positive\_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos\_format() and the remainder appear at the end of the formatted string.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1168 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_positive\_sign().

### 5.927.2.18 thousands\_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const [inline], [inherited]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning returning moneypunct<char↵\_type>::do\_thousands\_sep().

#### Returns

char\_type representing a thousands separator.

Definition at line 1108 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_thousands\_sep().

## 5.927.3 Member Data Documentation

### 5.927.3.1 id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 1043 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.928 std::move\_iterator< \_Iterator > Class Template Reference

### Public Types

- typedef \_\_traits\_type::difference\_type **difference\_type**
- typedef \_\_traits\_type::iterator\_category **iterator\_category**
- typedef \_Iterator **iterator\_type**
- typedef \_Iterator **pointer**
- typedef [conditional](#)< [is\\_reference](#)< \_\_base\_ref >::value, typename [remove\\_reference](#)< \_\_base\_ref >::type &&, \_\_base\_ref >::type **reference**
- typedef \_\_traits\_type::value\_type **value\_type**

## Public Member Functions

- `_GLIBCXX17_CONSTEXPR move_iterator` (iterator\_type \_\_i)
- `template<typename _Iter >`  
`_GLIBCXX17_CONSTEXPR move_iterator` (const `move_iterator`< \_Iter > &\_\_i)
- `_GLIBCXX17_CONSTEXPR iterator_type base` () const
- `_GLIBCXX17_CONSTEXPR reference operator*` () const
- `_GLIBCXX17_CONSTEXPR move_iterator operator+` (difference\_type \_\_n) const
- `_GLIBCXX17_CONSTEXPR move_iterator & operator++` ()
- `_GLIBCXX17_CONSTEXPR move_iterator operator++` (int)
- `_GLIBCXX17_CONSTEXPR move_iterator & operator+=` (difference\_type \_\_n)
- `_GLIBCXX17_CONSTEXPR move_iterator operator-` (difference\_type \_\_n) const
- `_GLIBCXX17_CONSTEXPR move_iterator & operator--` ()
- `_GLIBCXX17_CONSTEXPR move_iterator operator--` (int)
- `_GLIBCXX17_CONSTEXPR move_iterator & operator-=` (difference\_type \_\_n)
- `_GLIBCXX17_CONSTEXPR pointer operator->` () const
- `_GLIBCXX17_CONSTEXPR reference operator[]` (difference\_type \_\_n) const

## Protected Types

- `typedef __traits_type::reference __base_ref`
- `typedef iterator_traits< _Iterator > __traits_type`

## Protected Attributes

- `_Iterator _M_current`

## 5.928.1 Detailed Description

```
template<typename _Iterator>
class std::move_iterator< _Iterator >
```

Class template `move_iterator` is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 1007 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 5.929 `std::multimap< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

### Public Member Functions

- `multimap` ()=default
- `multimap` (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `multimap` (const `multimap` &)=default
- `multimap` (`multimap` &&)=default
- `multimap` (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `multimap` (const `allocator_type` &\_\_a)
- `multimap` (const `multimap` &\_\_m, const `allocator_type` &\_\_a)
- `multimap` (`multimap` &&\_\_m, const `allocator_type` &\_\_a) noexcept(`is_nothrow_copy_constructible`< `_Compare` >::value && `_Alloc_traits::S_always_equal`())
- `multimap` (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- template<typename `_InputIterator` >  
  `multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a)
- template<typename `_InputIterator` >  
  `multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` >  
  `multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `~multimap` ()=default
- iterator `begin` () noexcept
- const\_iterator `begin` () const noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- const\_reverse\_iterator `crbegin` () const noexcept
- const\_reverse\_iterator `crend` () const noexcept
- template<typename... `_Args` >  
  iterator `emplace` (`_Args` &&... \_\_args)

- template<typename... \_Args>  
iterator [emplace\\_hint](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool [empty](#) () const noexcept
  - iterator [end](#) () noexcept
  - const\_iterator [end](#) () const noexcept
  - size\_type [erase](#) (const key\_type &\_\_x)
  - iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type [get\\_allocator](#) () const noexcept
  - template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void [insert](#) (initializer\_list< value\_type > \_\_l)
  - key\_compare [key\\_comp](#) () const
  - size\_type [max\\_size](#) () const noexcept
  - multimap & [operator=](#) (const multimap &)=default
  - multimap & [operator=](#) (multimap &&)=default
  - multimap & [operator=](#) (initializer\_list< value\_type > \_\_l)
  - reverse\_iterator [rbegin](#) () noexcept
  - const\_reverse\_iterator [rbegin](#) () const noexcept
  - reverse\_iterator [rend](#) () noexcept
  - const\_reverse\_iterator [rend](#) () const noexcept
  - size\_type [size](#) () const noexcept
  - void [swap](#) (multimap &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare [value\\_comp](#) () const
- 
- iterator [insert](#) (const value\_type &\_\_x)
  - iterator [insert](#) (value\_type &&\_\_x)
  - template<typename \_Pair >  
\_\_enable\_if\_t< [is\\_constructible](#)< value\_type, \_Pair >::value, iterator > [insert](#) (\_Pair &&\_\_x)
- 
- iterator [insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator [insert](#) (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_Pair >  
\_\_enable\_if\_t< [is\\_constructible](#)< value\_type, \_Pair && >::value, iterator > [insert](#) (const\_iterator \_\_position, \_Pair &&\_\_x)
- 
- iterator [erase](#) (const\_iterator \_\_position)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator [erase](#) (iterator \_\_position)
- 
- iterator [find](#) (const key\_type &\_\_x)



- `template<typename _Kt >`  
`auto find (const _Kt &__x) -> decltype(_M.t._M_find_tr(__x))`
- `const_iterator find (const key_type &__x) const`
- `template<typename _Kt >`  
`auto find (const _Kt &__x) const -> decltype(_M.t._M_find_tr(__x))`
- `size_type count (const key_type &__x) const`
- `template<typename _Kt >`  
`auto count (const _Kt &__x) const -> decltype(_M.t._M_count_tr(__x))`
- `iterator lower_bound (const key_type &__x)`
- `template<typename _Kt >`  
`auto lower_bound (const _Kt &__x) -> decltype(iterator(_M.t._M_lower_bound_tr(__x)))`
- `const_iterator lower_bound (const key_type &__x) const`
- `template<typename _Kt >`  
`auto lower_bound (const _Kt &__x) const -> decltype(const_iterator(_M.t._M_lower_bound_tr(__x)))`
- `iterator upper_bound (const key_type &__x)`
- `template<typename _Kt >`  
`auto upper_bound (const _Kt &__x) -> decltype(iterator(_M.t._M_upper_bound_tr(__x)))`
- `const_iterator upper_bound (const key_type &__x) const`
- `template<typename _Kt >`  
`auto upper_bound (const _Kt &__x) const -> decltype(const_iterator(_M.t._M_upper_bound_tr(__x)))`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `template<typename _Kt >`  
`auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M.t._M_equal_range_tr(__x)))`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `template<typename _Kt >`  
`auto equal_range (const _Kt &__x) const -> decltype(pair< const_iterator, const_iterator >(_M.t._M_equal_range_tr(__x)))`

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**< (const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &, const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1 >  
bool **operator**== (const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &, const [multimap](#)< \_K1, \_T1, \_C1, \_A1 > &)

## 5.929.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >>>
class std::multimap< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key,T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 72 of file `stl_map.h`.

## 5.929.2 Constructor &amp; Destructor Documentation

5.929.2.1 `multimap()` [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap ( ) [default]
```

Default constructor creates no elements.

**5.929.2.2** `multimap()` [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a multimap with no elements.

**Parameters**

<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Definition at line 189 of file `stl_multimap.h`.

**5.929.2.3** `multimap()` [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    const multimap<_Key, _Tp, _Compare, _Alloc > & ) [default]
```

Multimap copy constructor.

Whether the allocator is copied depends on the allocator traits.

**5.929.2.4** `multimap()` [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    multimap<_Key, _Tp, _Compare, _Alloc > && ) [default]
```

Multimap move constructor.

The newly-created multimap contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multimap.

**5.929.2.5** `multimap()` [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
    initializer_list<value_type> & __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a multimap from an `initializer_list`.

## Parameters

<code>__l</code>	An initializer_list.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multimap consisting of copies of the elements from the initializer\_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 223 of file `stl_multimap.h`.

## 5.929.2.6 multimap() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    const allocator_type & __a ) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 231 of file `stl_multimap.h`.

## 5.929.2.7 multimap() [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    const multimap< _Key, _Tp, _Compare, _Alloc > & __m,
    const allocator_type & __a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 235 of file `stl_multimap.h`.

## 5.929.2.8 multimap() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    multimap< _Key, _Tp, _Compare, _Alloc > && __m,
    const allocator_type & __a ) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 239 of file `stl_multimap.h`.

**5.929.2.9** `multimap()` [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    initializer_list< value_type > __l,
    const allocator_type & __a ) [inline]
```

Allocator-extended initialier-list constructor.

Definition at line 245 of file `stl_multimap.h`.

**5.929.2.10** `multimap()` [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a ) [inline]
```

Allocator-extended range constructor.

Definition at line 251 of file `stl_multimap.h`.

**5.929.2.11** `multimap()` [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Builds a multimap from a range.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a multimap consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 267 of file `stl_multimap.h`.

## 5.929.2.12 multimap() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a multimap from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 283 of file `stl_multimap.h`.

## 5.929.2.13 ~multimap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::~~multimap ( ) [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 5.929.3 Member Function Documentation

## 5.929.3.1 begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 349 of file `stl_multimap.h`.

#### 5.929.3.2 begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 358 of file stl\_multimap.h.

#### 5.929.3.3 cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 422 of file stl\_multimap.h.

#### 5.929.3.4 cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 431 of file stl\_multimap.h.

#### 5.929.3.5 clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap<_Key, _Tp, _Compare, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.



Definition at line 806 of file stl\_multimap.h.

#### 5.929.3.6 count() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap<_Key, _Tp, _Compare, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements with given key.

## Parameters

<a href="#"></a>	Key of (key, value) pairs to be located.
<a href="#"></a>	

## Returns

Number of elements with specified key.



Definition at line 883 of file stl\_multimap.h.

## 5.929.3.7 count() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
    const _Kt & __x ) const -> decltype(_M_t._M_count_tr(__x))    [inline]
```

Finds the number of elements with given key.

## Parameters

<a href="#"></a>	Key of (key, value) pairs to be located.
<a href="#"></a>	

## Returns

Number of elements with specified key.

Definition at line 889 of file stl\_multimap.h.

## 5.929.3.8 crbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin ( ) const    [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 440 of file stl\_multimap.h.



#### 5.929.3.9 `crend()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 449 of file `stl_multimap.h`.

#### 5.929.3.10 `emplace()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Build and insert a `std::pair` into the multimap.

##### Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

##### Returns

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 489 of file `stl_multimap.h`.

#### 5.929.3.11 `emplace_hint()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Builds and inserts a `std::pair` into the multimap.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

## Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 516 of file `stl_multimap.h`.

5.929.3.12 `empty()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the multimap is empty.

Definition at line 456 of file `stl_multimap.h`.

5.929.3.13 `end()` [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 367 of file `stl_multimap.h`.

**5.929.3.14** `end()` [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 376 of file `stl_multimap.h`.

**5.929.3.15** `equal_range()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, iterator> std::multimap<_Key, _Tp, _Compare, _Alloc >::equal_range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 999 of file `stl_multimap.h`.

**5.929.3.16** `equal_range()` [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap<_Key, _Tp, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

## Parameters

<code>_↔</code>	Key of (key, value) pairs to be located.
<code>_X</code>	

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1005 of file stl\_multimap.h.

## 5.929.3.17 equal\_range() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<const_iterator, const_iterator> std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_↔
range (
    const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>_↔</code>	Key of (key, value) pairs to be located.
<code>_X</code>	

## Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1026 of file stl\_multimap.h.

**5.929.3.18** `equal_range()` [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) const -> decltype(pair<const_iterator, const_iterator> ( _M_t._M_↵
equal_range_tr(__x)))    [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>↵</code> <code>__x</code>	Key of (key, value) pairs to be located.
------------------------------------	--

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1032 of file `stl_multimap.h`.

**5.929.3.19** `erase()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
    const_iterator __position ) [inline]
```

Erases an element from a multimap.

**Parameters**

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 700 of file stl\_multimap.h.

#### 5.929.3.20 erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
    iterator __position ) [inline]
```

Erases an element from a multimap.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

##### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 706 of file stl\_multimap.h.

#### 5.929.3.21 erase() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

##### Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 737 of file `stl_multimap.h`.

**5.929.3.22 erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a `[first,last)` range of elements from a multimap.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased .

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 758 of file `stl_multimap.h`.

**5.929.3.23 find()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in a multimap.

## Parameters

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 841 of file `stl_multimap.h`.

## 5.929.3.24 find() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x ) -> decltype(_M_t._M_find_tr(__x))    [inline]
```

Tries to locate an element in a multimap.

## Parameters

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 847 of file `stl_multimap.h`.

## 5.929.3.25 find() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const key_type & __x ) const    [inline]
```

Tries to locate an element in a multimap.



**Parameters**

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

**Returns**

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 865 of file `stl_multimap.h`.

**5.929.3.26 find()** [ 4 / 4 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
    const _Kt & __x ) const -> decltype(_M_t._M_find_tr(__x))    [inline]
```

Tries to locate an element in a multimap.

**Parameters**

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

**Returns**

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 871 of file `stl_multimap.h`.

**5.929.3.27 get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator ( ) const    [inline],
[noexcept]
```

Get a copy of the memory allocation object.


Definition at line 339 of file `stl_multimap.h`.

**5.929.3.28** insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

 _x	Pair to be inserted (see std::make_pair for easy creation of pairs).
--	--

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 537 of file stl\_multimap.h.


Referenced by std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

**5.929.3.29** insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
    value_type && __x ) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

 _x	Pair to be inserted (see std::make_pair for easy creation of pairs).
--	--

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 544 of file stl\_multimap.h.

#### 5.929.3.30 insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair>::value, iterator> std::multimap<_Key, _Tp, ↵
_Compare, _Alloc >::insert (
    _Pair && __x ) [inline]
```

Inserts a std::pair into the multimap.

##### Parameters

<code>↵</code> <code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).
------------------------------------	--

##### Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 549 of file stl\_multimap.h.

#### 5.929.3.31 insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x ) [inline]
```

Inserts a std::pair into the multimap.

##### Parameters

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 577 of file stl\_multimap.h.

**5.929.3.32 insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    const_iterator __position,
    value_type && __x ) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 587 of file stl\_multimap.h.

**5.929.3.33** `insert()` [ 6/8 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::multimap<_Key, _Tp,
_Compare, _Alloc >::insert (
    const_iterator __position,
    _Pair && __x ) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 592 of file `stl_multimap.h`.

**5.929.3.34** `insert()` [ 7/8 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
void std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 611 of file stl\_multimap.h.

#### 5.929.3.35 insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of std::pairs into the multimap.

##### Parameters

↔	A std::initializer_list<value_type> of pairs to be inserted.
↔	
↔	
↔	
/	

Complexity similar to that of the range constructor.

Definition at line 623 of file stl\_multimap.h.

References std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

#### 5.929.3.36 key\_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

Definition at line 815 of file stl\_multimap.h.

#### 5.929.3.37 lower\_bound() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 907 of file `stl_multimap.h`.

**5.929.3.38 lower\_bound()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) -> decltype(iterator(_M.t._M_lower_bound_tr(__x)))    [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>_↵</code>	Key of (key, value) pair to be located.
<code>_X</code>	

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 913 of file `stl_multimap.h`.

**5.929.3.39 lower\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) const    [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

$\leftarrow$ _X	Key of (key, value) pair to be located.
--------------------	---

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 932 of file stl\_multimap.h.

## 5.929.3.40 lower\_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

$\leftarrow$ _X	Key of (key, value) pair to be located.
--------------------	---

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 938 of file stl\_multimap.h.

## 5.929.3.41 max\_size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the maximum size of the multimap.

Definition at line 466 of file stl\_multimap.h.



**5.929.3.42 operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap& std::multimap<_Key, _Tp, _Compare, _Alloc >::operator= (
    const multimap<_Key, _Tp, _Compare, _Alloc > & ) [default]
```

Multimap assignment operator.

Whether the allocator is copied depends on the allocator traits.

**5.929.3.43 operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap& std::multimap<_Key, _Tp, _Compare, _Alloc >::operator= (
    multimap<_Key, _Tp, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**5.929.3.44 operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap& std::multimap<_Key, _Tp, _Compare, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Multimap list assignment operator.

**Parameters**

↩	An initializer_list.
↩	
↩	
↩	
/	

This function fills a multimap with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned.

Definition at line 330 of file stl\_multimap.h.

**5.929.3.45 rbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >>  
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 385 of file stl\_multimap.h.

**5.929.3.46 rbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >>  
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) const [inline],  
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 394 of file stl\_multimap.h.

**5.929.3.47 rend()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >>  
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 403 of file stl\_multimap.h.

**5.929.3.48 rend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =  
std::allocator<std::pair<const _Key, _Tp> >>  
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend ( ) const [inline],  
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 412 of file stl\_multimap.h.

**5.929.3.49 size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap<_Key, _Tp, _Compare, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the size of the multimap.

Definition at line 461 of file `stl_multimap.h`.

**5.929.3.50 swap()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap<_Key, _Tp, _Compare, _Alloc >::swap (
    multimap<_Key, _Tp, _Compare, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another multimap.

**Parameters**

<b><code>__x</code></b>	A multimap of the same element and allocator types.
-------------------------	---

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 795 of file `stl_multimap.h`.

**5.929.3.51 upper\_bound()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

<b><code>__x</code></b>	Key of (key, value) pair to be located.
-------------------------	---

**Returns**

Iterator pointing to the first element greater than key, or end().



Definition at line 952 of file stl\_multimap.h.

**5.929.3.52 upper\_bound()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

	Key of (key, value) pair to be located.
	

**Returns**

Iterator pointing to the first element greater than key, or end().



Definition at line 958 of file stl\_multimap.h.

**5.929.3.53 upper\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) const    [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

	Key of (key, value) pair to be located.
	

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 972 of file `stl_multimap.h`.

#### 5.929.3.54 `upper_bound()` [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

##### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or `end()`.

Definition at line 978 of file `stl_multimap.h`.

#### 5.929.3.55 `value_comp()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

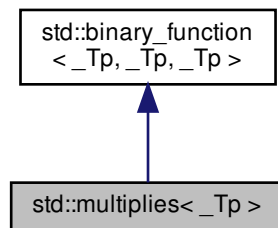
Definition at line 823 of file `stl_multimap.h`.

The documentation for this class was generated from the following files:

- [stl\\_map.h](#)
- [stl\\_multimap.h](#)

## 5.930 std::multiplies< \_Tp > Struct Template Reference

Inheritance diagram for std::multiplies< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `_GLIBCXX14_CONSTEXPR _Tp operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.930.1 Detailed Description

```
template<typename _Tp>  
struct std::multiplies< _Tp >
```

One of the [math functors](#).

Definition at line 153 of file `stl_function.h`.

#### 5.930.2 Member Typedef Documentation

#### 5.930.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.930.2.2 result\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.930.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.931 std::multiplies< void > Struct Template Reference

#### Public Types

- typedef \_\_is\_transparent **is\_transparent**

#### Public Member Functions

- template<typename \_Tp, typename \_Up >  
\_GLIBCXX14\_CONSTEXPR auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward<\_Tp>(\_\_t) \*std::forward<\_Up>(\_\_u))) -> decltype(std::forward<\_Tp>(\_\_t) \*std::forward<\_Up>(\_\_u))

## 5.931.1 Detailed Description

```
template<>
struct std::multiplies< void >
```

One of the [math functors](#).

Definition at line 260 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

5.932 `std::multiset< _Key, _Compare, _Alloc >` Class Template Reference

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::const_iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::const_reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- [multiset](#) ()=default
- [multiset](#) (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- template<typename `_InputIterator` >  
[multiset](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` >  
[multiset](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- [multiset](#) (const [multiset](#) &)=default
- [multiset](#) ([multiset](#) &&)=default
- [multiset](#) ([initializer\\_list](#)< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- [multiset](#) (const `allocator_type` &\_\_a)
- [multiset](#) (const [multiset](#) &\_\_m, const `allocator_type` &\_\_a)



- `multiset` (`multiset` &&\_\_m, const allocator\_type &\_\_a) noexcept(`is_nothrow_copy_constructible`< \_Compare >↵  
::value &&\_Alloc\_traits::S\_always\_equal())
  - `multiset` (`initializer_list`< value\_type > \_\_l, const allocator\_type &\_\_a)
  - template<typename \_InputIterator >  
`multiset` (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
  - `~multiset` ()=default
  - iterator `begin` () const noexcept
  - iterator `cbegin` () const noexcept
  - iterator `cend` () const noexcept
  - void `clear` () noexcept
  - `reverse_iterator` `crbegin` () const noexcept
  - `reverse_iterator` `crend` () const noexcept
  - template<typename... \_Args>  
iterator `emplace` (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator `emplace_hint` (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool `empty` () const noexcept
  - iterator `end` () const noexcept
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator `erase` (const\_iterator \_\_position)
  - size\_type `erase` (const key\_type &\_\_x)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type `get_allocator` () const noexcept
  - iterator `insert` (const value\_type &\_\_x)
  - iterator `insert` (value\_type &&\_\_x)
  - iterator `insert` (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void `insert` (`initializer_list`< value\_type > \_\_l)
  - key\_compare `key_comp` () const
  - size\_type `max_size` () const noexcept
  - `multiset` & `operator=` (const `multiset` &)=default
  - `multiset` & `operator=` (`multiset` &&)=default
  - `multiset` & `operator=` (`initializer_list`< value\_type > \_\_l)
  - `reverse_iterator` `rbegin` () const noexcept
  - `reverse_iterator` `rend` () const noexcept
  - size\_type `size` () const noexcept
  - void `swap` (`multiset` &\_\_x) noexcept(`/*conditional */`)
  - value\_compare `value_comp` () const
- 
- size\_type `count` (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto `count` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- 
- iterator `find` (const key\_type &\_\_x)

- const\_iterator [find](#) (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto [find](#) (const \_Kt &\_\_x) -> decltype(iterator
  - template<typename \_Kt >  
auto [find](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator
- 
- iterator [lower\\_bound](#) (const key\_type &\_\_x)
  - const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
  - template<typename \_Kt >  
auto [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- 
- iterator [upper\\_bound](#) (const key\_type &\_\_x)
  - const\_iterator [upper\\_bound](#) (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto [upper\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
  - template<typename \_Kt >  
auto [upper\\_bound](#) (const \_Kt &\_\_x) const -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- 
- std::pair< iterator, iterator > [equal\\_range](#) (const key\_type &\_\_x)
  - std::pair< const\_iterator, const\_iterator > [equal\\_range](#) (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto [equal\\_range](#) (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))
  - template<typename \_Kt >  
auto [equal\\_range](#) (const \_Kt &\_\_x) const -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))

#### Friends

- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator**< (const [multiset](#)< \_K1, \_C1, \_A1 > &, const [multiset](#)< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator**== (const [multiset](#)< \_K1, \_C1, \_A1 > &, const [multiset](#)< \_K1, \_C1, \_A1 > &)

#### 5.932.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::multiset< _Key, _Compare, _Alloc >
```

A standard container made up of elements, which can be retrieved in logarithmic time.

### Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for `set` and `multiset`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 96 of file `stl_multiset.h`.

## 5.932.2 Constructor & Destructor Documentation

### 5.932.2.1 `multiset()` [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset ( ) [default]
```

Default constructor creates no elements.

### 5.932.2.2 `multiset()` [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a multiset with no elements.

### Parameters

<code>__comp</code>	Comparator to use.
<code>__a</code>	An allocator object.

Definition at line 173 of file `stl_multiset.h`.

## 5.932.2.3 multiset() [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::multiset<_Key, _Compare, _Alloc >::multiset (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Builds a multiset from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(\_\_first,\_\_last)).

Definition at line 187 of file stl\_multiset.h.

## 5.932.2.4 multiset() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::multiset<_Key, _Compare, _Alloc >::multiset (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a multiset from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multiset consisting of copies of the elements from [\_\_first,\_\_last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(\_\_first,\_\_last)).

Definition at line 203 of file stl\_multiset.h.

**5.932.2.5 multiset()** [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    const multiset< _Key, _Compare, _Alloc > & ) [default]
```

Multiset copy constructor.

Whether the allocator is copied depends on the allocator traits.

**5.932.2.6 multiset()** [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    multiset< _Key, _Compare, _Alloc > && ) [default]
```

Multiset move constructor.

The newly-created multiset contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multiset.

**5.932.2.7 multiset()** [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    initializer_list< value_type > __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a multiset from an initializer\_list.

**Parameters**

<code>__l</code>	An initializer_list.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 239 of file `stl_multiset.h`.

**5.932.2.8 multiset()** [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>>
```

```
std::multiset< _Key, _Compare, _Alloc >::multiset (
    const allocator_type & __a ) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 247 of file stl\_multiset.h.

#### 5.932.2.9 multiset() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    const multiset< _Key, _Compare, _Alloc > & __m,
    const allocator_type & __a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 251 of file stl\_multiset.h.

#### 5.932.2.10 multiset() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    multiset< _Key, _Compare, _Alloc > && __m,
    const allocator_type & __a ) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 255 of file stl\_multiset.h.

#### 5.932.2.11 multiset() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
    initializer_list< value_type > __l,
    const allocator_type & __a ) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 261 of file stl\_multiset.h.

**5.932.2.12** `multiset()` [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _InputIterator >
std::multiset< _Key, _Compare, _Alloc >::multiset (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a ) [inline]
```

Allocator-extended range constructor.

Definition at line 267 of file `stl_multiset.h`.

**5.932.2.13** `~multiset()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
std::multiset< _Key, _Compare, _Alloc >::~~multiset ( ) [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**5.932.3** Member Function Documentation**5.932.3.1** `begin()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
iterator std::multiset< _Key, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 340 of file `stl_multiset.h`.

**5.932.3.2** `cbegin()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
iterator std::multiset< _Key, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 377 of file `stl_multiset.h`.

### 5.932.3.3 cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 386 of file stl\_multiset.h.

### 5.932.3.4 clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 718 of file stl\_multiset.h.

### 5.932.3.5 count() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements with given key.

#### Parameters

<code>__x</code>	Key of elements to be located.
------------------	--------------------------------

#### Returns

Number of elements with specified key.

Definition at line 730 of file stl\_multiset.h.



**5.932.3.6 count()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::count (
    const _Kt & __x ) const -> decltype(_M_t._M_count_tr(__x))    [inline]
```

Finds the number of elements with given key.

**Parameters**

<code>__x</code>	Key of elements to be located.
------------------	--------------------------------

**Returns**

Number of elements with specified key.

Definition at line 736 of file `stl_multiset.h`.

**5.932.3.7 crbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crbegin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 395 of file `stl_multiset.h`.

**5.932.3.8 crend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crend ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 404 of file `stl_multiset.h`.

**5.932.3.9 emplace()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
iterator std::multiset< _Key, _Compare, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Builds and inserts an element into the multiset.

## Parameters

<code>__args</code>	Arguments used to generate the element instance to be inserted.
---------------------	---

## Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 457 of file `stl_multiset.h`.

5.932.3.10 `emplace_hint()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
iterator std::multiset< _Key, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Builds and inserts an element into the multiset.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element instance to be inserted.

## Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 483 of file `stl_multiset.h`.

### 5.932.3.11 empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 410 of file stl\_multiset.h.

### 5.932.3.12 end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 349 of file stl\_multiset.h.

### 5.932.3.13 equal\_range() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::pair<iterator, iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

#### Parameters

<code>↵</code> <code>__x</code>	Key to be located.
------------------------------------	--------------------

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 860 of file stl\_multiset.h.

## 5.932.3.14 equal\_range() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::multiset<_Key, _Compare, _Alloc >::equal_range (
    const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 864 of file stl\_multiset.h.

## 5.932.3.15 equal\_range() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset<_Key, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 870 of file stl\_multiset.h.

#### 5.932.3.16 equal\_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
template<typename _Kt >  
auto std::multiset< _Key, _Compare, _Alloc >::equal_range (   
    const _Kt & __x ) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_↵  
tr(__x))) [inline]
```

Finds a subsequence matching given key.

##### Parameters

<code>↵</code>	Key to be located.
<code>__x</code>	

##### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 876 of file stl\_multiset.h.

#### 5.932.3.17 erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (   
    const_iterator __position ) [inline]
```

Erases an element from a multiset.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 639 of file `stl_multiset.h`.

## 5.932.3.18 erase() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::erase (
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

## Parameters

<code>↵</code> <code>__x</code>	Key of element to be erased.
------------------------------------	------------------------------

## Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 669 of file `stl_multiset.h`.

## 5.932.3.19 erase() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a `[first,last)` range of elements from a multiset.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

**Returns**

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 691 of file `stl_multiset.h`.

**5.932.3.20 find()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in a set.

**Parameters**

<code>↵</code>	Element to be located.
<code>__x</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 756 of file `stl_multiset.h`.

**5.932.3.21 find()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in a set.

## Parameters

<code>_↵</code>	Element to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 760 of file `stl_multiset.h`.

## 5.932.3.22 find() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::find (
    const _Kt & __x ) -> decltype(iterator [inline]
```

Tries to locate an element in a set.

## Parameters

<code>_↵</code>	Element to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 766 of file `stl_multiset.h`.

## 5.932.3.23 find() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::find (
    const _Kt & __x ) const -> decltype(const_iterator [inline]
```

Tries to locate an element in a set.



**Parameters**

<code>_↵</code>	Element to be located.
<code>_X</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 772 of file `stl_multiset.h`.

**5.932.3.24 `get_allocator()`**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]
```

Returns the memory allocation object.

Definition at line 331 of file `stl_multiset.h`.

**5.932.3.25 `insert()`** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Inserts an element into the multiset.

**Parameters**

<code>_↵</code>	Element to be inserted.
<code>_X</code>	

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 502 of file stl\_multiset.h.

Referenced by std::multiset< \_Key, \_Compare, \_Alloc >::insert().

#### 5.932.3.26 insert() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x ) [inline]
```

Inserts an element into the multiset.

##### Parameters

<code>__position</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

##### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 532 of file stl\_multiset.h.

#### 5.932.3.27 insert() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _InputIterator >
void std::multiset< _Key, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that tries to insert a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 551 of file `stl_multiset.h`.

**5.932.3.28 insert()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::multiset< _Key, _Compare, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the multiset.

**Parameters**

<code>↵</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
<code>↵</code>	
<code>↵</code>	
<code>↵</code>	
<code>/</code>	

Complexity similar to that of the range constructor.

Definition at line 563 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::insert()`.

**5.932.3.29 key\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp ( ) const [inline]
```

Returns the comparison object.

Definition at line 323 of file `stl_multiset.h`.

**5.932.3.30 lower\_bound()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

$\_k$	Key to be located.
$\_x$	

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 791 of file stl\_multiset.h.

## 5.932.3.31 lower\_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

$\_k$	Key to be located.
$\_x$	

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 795 of file stl\_multiset.h.

## 5.932.3.32 lower\_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) -> decltype(iterator(_M.t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

$\_key$	Key to be located.
$\_x$	

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 801 of file stl\_multiset.h.

**5.932.3.33 lower\_bound()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) const -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))    [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

$\_key$	Key to be located.
$\_x$	

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 807 of file stl\_multiset.h.

**5.932.3.34 max\_size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
size_type std::multiset< _Key, _Compare, _Alloc >::max_size ( ) const    [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 420 of file stl\_multiset.h.

**5.932.3.35 operator=()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset<_Key, _Compare, _Alloc >::operator= (
    const multiset<_Key, _Compare, _Alloc > & ) [default]
```

Multiset assignment operator.

Whether the allocator is copied depends on the allocator traits.

**5.932.3.36 operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset<_Key, _Compare, _Alloc >::operator= (
    multiset<_Key, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**5.932.3.37 operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset<_Key, _Compare, _Alloc >::operator= (
    initializer_list<value_type > __l ) [inline]
```

Multiset list assignment operator.

**Parameters**

↩	An initializer_list.
↩	
↩	
↩	
/	

This function fills a multiset with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned.

Definition at line 312 of file stl\_multiset.h.

**5.932.3.38 rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 358 of file stl\_multiset.h.

**5.932.3.39 rend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 367 of file stl\_multiset.h.

**5.932.3.40 size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the size of the set.

Definition at line 415 of file stl\_multiset.h.

**5.932.3.41 swap()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::swap (
    multiset< _Key, _Compare, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another multiset.

**Parameters**

<code>↵</code> <code>__x</code>	A multiset of the same element and allocator types.
------------------------------------	---

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 437 of file `stl_multiset.h`.

#### 5.932.3.42 upper\_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

<code>↵</code> <code>__x</code>	Key to be located.
------------------------------------	--------------------

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 821 of file `stl_multiset.h`.

#### 5.932.3.43 upper\_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

<code>↵</code> <code>__x</code>	Key to be located.
------------------------------------	--------------------

##### Returns

Iterator pointing to the first element greater than key, or `end()`.



Definition at line 825 of file `stl_multiset.h`.

#### 5.932.3.44 `upper_bound()` [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 831 of file `stl_multiset.h`.

#### 5.932.3.45 `upper_bound()` [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 837 of file `stl_multiset.h`.

## 5.932.3.46 value\_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::multiset< _Key, _Compare, _Alloc >::value_comp ( ) const [inline]
```

Returns the comparison object.

Definition at line 327 of file `stl_multiset.h`.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

## 5.933 std::mutex Class Reference

Inherits `std::__mutex_base`.

## Public Types

- typedef `__native_type` \* **native\_handle\_type**

## Public Member Functions

- **mutex** (const [mutex](#) &)=delete
- void **lock** ()
- `native_handle_type` **native\_handle** () noexcept
- [mutex](#) & **operator=** (const [mutex](#) &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

## Private Types

- typedef `__gthread_mutex_t` **\_\_native\_type**

## Private Attributes

- `__native_type` **\_M\_mutex**

## 5.933.1 Detailed Description

The standard mutex type.

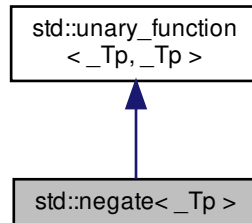
Definition at line 86 of file `std_mutex.h`.

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

### 5.934 std::negate<\_Tp> Struct Template Reference

Inheritance diagram for std::negate<\_Tp>:



#### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Tp` [result\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR _Tp operator() (const _Tp &__x) const`

#### 5.934.1 Detailed Description

```
template<typename _Tp>
struct std::negate<_Tp>
```

One of the [math functors](#).

Definition at line 162 of file `stl_function.h`.

#### 5.934.2 Member Typedef Documentation

##### 5.934.2.1 `argument_type`

```
typedef _Tp std::unary\_function<\_Tp, \_Tp>::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

5.934.2.2 `result_type`

```
typedef _Tp std::unary\_function< _Tp , _Tp >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

5.935 `std::negate< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **`is_transparent`**

## Public Member Functions

- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR auto operator() (_Tp &&__t) const noexcept(noexcept(-std::forward< _Tp >(__t)))`  
`-> decltype(-std::forward< _Tp >(__t))`

## 5.935.1 Detailed Description

```
template<>
struct std::negate< void >
```

One of the [math functors](#).

Definition at line 305 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

5.936 `std::negative_binomial_distribution< _IntType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- **negative\_binomial\_distribution** (`_IntType __k=1, double __p=0.5`)
- **negative\_binomial\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_IntType k` () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- double [p](#) () const
- [param\\_type param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >  
[std::basic\\_ostream](#)< `_CharT`, `_Traits` > & [operator<<](#) ([std::basic\\_ostream](#)< `_CharT`, `_Traits` > &\_\_os, const [std::negative\\_binomial\\_distribution](#)< `_IntType1` > &\_\_x)
- bool [operator==](#) (const [negative\\_binomial\\_distribution](#) &\_\_d1, const [negative\\_binomial\\_distribution](#) &\_\_d2)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >  
[std::basic\\_istream](#)< `_CharT`, `_Traits` > & [operator>>](#) ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &\_\_is, [std::negative\\_binomial\\_distribution](#)< `_IntType1` > &\_\_x)

## 5.936.1 Detailed Description

```
template<typename _IntType = int>
class std::negative_binomial_distribution< _IntType >
```

A `negative_binomial_distribution` random number distribution.

The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 4103 of file `random.h`.

### 5.936.2 Member Typedef Documentation

#### 5.936.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::negative_binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 4106 of file random.h.

### 5.936.3 Member Function Documentation

#### 5.936.3.1 k()

```
template<typename _IntType = int>
_IntType std::negative_binomial_distribution< _IntType >::k ( ) const [inline]
```

Return the  $k$  parameter of the distribution.

Definition at line 4166 of file random.h.

#### 5.936.3.2 max()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4202 of file random.h.

References std::numeric\_limits< \_Tp >::max().

#### 5.936.3.3 min()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4195 of file random.h.

#### 5.936.3.4 operator()

```
template<typename _IntType >
template<typename _UniformRandomNumberGenerator >
negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _↵
_IntType >::operator() (
    _UniformRandomNumberGenerator & __urng )
```

Generating functions.

Definition at line 1123 of file bits/random.tcc.

#### 5.936.3.5 p()

```
template<typename _IntType = int>
double std::negative_binomial_distribution< _IntType >::p ( ) const [inline]
```

Return the  $p$  parameter of the distribution.

Definition at line 4173 of file random.h.

#### 5.936.3.6 param() [1/2]

```
template<typename _IntType = int>
param_type std::negative_binomial_distribution< _IntType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4180 of file random.h.

#### 5.936.3.7 param() [2/2]

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4188 of file random.h.

### 5.936.3.8 reset()

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 4159 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

## 5.936.4 Friends And Related Function Documentation

### 5.936.4.1 operator<<

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::negative_binomial_distribution< _IntType1 > & __x ) [friend]
```

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>negative_binomial_distribution</code> random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

### 5.936.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
    const negative_binomial_distribution< _IntType > & __d1,
    const negative_binomial_distribution< _IntType > & __d2 ) [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4251 of file random.h.



5.936.4.3 `operator>>`

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::negative_binomial_distribution< _IntType1 > & __x ) [friend]
```

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>negative_binomial_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.937 `std::negative_binomial_distribution<_IntType>::param_type` Struct Reference

## Public Types

- typedef `negative_binomial_distribution<_IntType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_IntType __k=1, double __p=0.5`)
- `_IntType k () const`
- `double p () const`

## Friends

- `bool operator!= (const param\_type &__p1, const param\_type &__p2)`
- `bool operator== (const param\_type &__p1, const param\_type &__p2)`

## 5.937.1 Detailed Description

```
template<typename _IntType = int>
struct std::negative_binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4113 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.938 `std::nested_exception` Class Reference

Inherited by `std::_Nested_exception< _Except >`.

## Public Member Functions

- **nested\_exception** (const [nested\\_exception](#) &) noexcept=default
- **exception\_ptr nested\_ptr** () const noexcept
- **nested\_exception & operator=** (const [nested\\_exception](#) &) noexcept=default
- void **rethrow\_nested** () const

## 5.938.1 Detailed Description

Exception class with `exception_ptr` data member.

Definition at line 52 of file `nested_exception.h`.

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

5.939 `std::normal_distribution< _RealType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [normal\\_distribution](#) ([result\\_type](#) \_\_mean=[result\\_type](#)(0), [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- **normal\_distribution** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- `template<typename _UniformRandomNumberGenerator >`  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [\\_RealType](#) **mean** () const
- [result\\_type](#) **min** () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- [\\_RealType](#) **stddev** () const

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::normal\\_distribution](#)< \_RealType1 > &\_\_x)
- `template<typename _RealType1 >`  
bool **operator==** (const [std::normal\\_distribution](#)< \_RealType1 > &\_\_d1, const [std::normal\\_distribution](#)< \_RealType1 > &\_\_d2)
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::normal\\_distribution](#)< \_RealType1 > &\_\_x)

## 5.939.1 Detailed Description

```
template<typename _RealType = double>
class std::normal_distribution< _RealType >
```

A normal continuous distribution for random numbers.

The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 1925 of file random.h.

### 5.939.2 Member Typedef Documentation

#### 5.939.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::normal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 1928 of file random.h.

### 5.939.3 Constructor & Destructor Documentation

#### 5.939.3.1 normal\_distribution()

```
template<typename _RealType = double>
std::normal_distribution< _RealType >::normal_distribution (
    result_type __mean = result_type(0),
    result_type __stddev = result_type(1) ) [inline], [explicit]
```

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 1975 of file random.h.

### 5.939.4 Member Function Documentation

#### 5.939.4.1 max()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2032 of file random.h.

#### 5.939.4.2 mean()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::mean ( ) const [inline]
```

Returns the mean of the distribution.

Definition at line 1996 of file random.h.

#### 5.939.4.3 min()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2025 of file random.h.

#### 5.939.4.4 operator>() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::normal_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 2040 of file random.h.

Referenced by std::normal\_distribution< result\_type >::operator>()().

#### 5.939.4.5 operator>() [2/2]

```
template<typename _RealType >
template<typename _UniformRandomNumberGenerator >
normal_distribution< _RealType >::result_type std::normal_distribution< _RealType >::operator()
(
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param )
```

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1786 of file bits/random.tcc.

## 5.939.4.6 param() [1/2]

```
template<typename _RealType = double>
param_type std::normal_distribution<_RealType>::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2010 of file random.h.

## 5.939.4.7 param() [2/2]

```
template<typename _RealType = double>
void std::normal_distribution<_RealType>::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

## Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2018 of file random.h.

## 5.939.4.8 reset()

```
template<typename _RealType = double>
void std::normal_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 1989 of file random.h.

Referenced by `std::lognormal_distribution<_RealType>::reset()`, `std::gamma_distribution<result_type>::reset()`, `std::student_t_distribution<_RealType>::reset()`, `std::binomial_distribution<_IntType>::reset()`, and `std::poisson_distribution<_IntType>::reset()`.

## 5.939.4.9 stddev()

```
template<typename _RealType = double>
_RealType std::normal_distribution<_RealType>::stddev ( ) const [inline]
```

Returns the standard deviation of the distribution.

Definition at line 2003 of file random.h.

## 5.939.5 Friends And Related Function Documentation

### 5.939.5.1 operator<<

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::normal_distribution< _RealType1 > & __x ) [friend]
```

Inserts a normal\_distribution random number distribution \_\_x into the output stream \_\_os.

#### Parameters

__os	An output stream.
__x	A normal_distribution random number distribution.

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

### 5.939.5.2 operator==

```
template<typename _RealType = double>
template<typename _RealType1 >
bool operator== (
    const std::normal_distribution< _RealType1 > & __d1,
    const std::normal_distribution< _RealType1 > & __d2 ) [friend]
```

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

### 5.939.5.3 operator>>

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::normal_distribution< _RealType1 > & __x ) [friend]
```

Extracts a normal\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>normal_distribution</code> random number generator engine.

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.940 `std::normal_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef [normal\\_distribution<\\_RealType>](#) **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __mean=_RealType(0), _RealType __stddev=_RealType(1)`)
- `_RealType mean` () const
- `_RealType stddev` () const

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.940.1 Detailed Description

```
template<typename _RealType = double>
struct std::normal_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 1935 of file `random.h`.

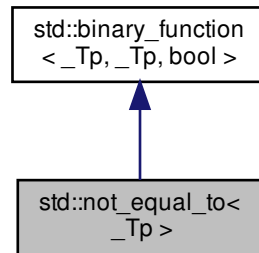
The documentation for this struct was generated from the following file:

- [random.h](#)



### 5.941 `std::not_equal_to<_Tp>` Struct Template Reference

Inheritance diagram for `std::not_equal_to<_Tp>`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR` `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 5.941.1 Detailed Description

```
template<typename _Tp>
struct std::not_equal_to<_Tp>
```

One of the [comparison functors](#).

Definition at line 334 of file `stl_function.h`.

#### 5.941.2 Member Typedef Documentation

## 5.941.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

## 5.941.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 5.941.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.942 std::not\_equal\_to&lt; void &gt; Struct Template Reference

## Public Types

- typedef \_\_is\_transparent **is\_transparent**

## Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) !=std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) !=std::forward< \_Up >(\_\_u))

#### 5.942.1 Detailed Description

```
template<>
struct std::not_equal_to< void >
```

One of the [comparison functors](#).

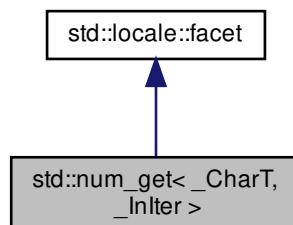
Definition at line 478 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 5.943 std::num\_get< \_CharT, \_InIter > Class Template Reference

Inheritance diagram for std::num\_get< \_CharT, \_InIter >:



#### Public Types

- typedef \_CharT [char\\_type](#)
- typedef \_InIter [iter\\_type](#)

## Public Member Functions

- [num\\_get](#) (size\_t \_\_refs=0)
- [template<typename \\_ValueT > \\_GLIBCXX\\_DEFAULT\\_ABI\\_TAG \\_InIter \\_M\\_extract\\_int](#) (\_InIter \_\_beg, \_InIter \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, \_ValueT &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, bool &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, void \*&\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, long &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, unsigned short &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, unsigned int &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, unsigned long &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, long long &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, unsigned long long &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, float &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, double &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, long double &\_\_v) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~num\\_get](#) ()
- [\\_GLIBCXX\\_DEFAULT\\_ABI\\_TAG iter\\_type \\_M\\_extract\\_float](#) ([iter\\_type](#), [iter\\_type](#), [ios\\_base](#) &, [ios\\_base::iostate](#) &, [string](#) &) const
- [template<typename \\_ValueT > \\_GLIBCXX\\_DEFAULT\\_ABI\\_TAG iter\\_type \\_M\\_extract\\_int](#) ([iter\\_type](#), [iter\\_type](#), [ios\\_base](#) &, [ios\\_base::iostate](#) &, \_ValueT &) const
- [template<typename \\_CharT2 > \\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, int >::\\_\\_type \\_M\\_find](#) (const \_CharT2 \*, size\_t \_\_len, \_CharT2 \_\_c) const
- [template<typename \\_CharT2 > \\_\\_gnu\\_cxx::\\_\\_enable\\_if<! \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, int >::\\_\\_type \\_M\\_find](#) (const \_CharT2 \* \_\_zero, size\_t \_\_len, \_CharT2 \_\_c) const

- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const`

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

#### 5.943.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >
```

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.

The `num_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_get` facet.

Definition at line 1952 of file `locale_facets.h`.

#### 5.943.2 Member Typedef Documentation

#### 5.943.2.1 char\_type

```
template<typename _CharT , typename _InIter >
typedef _CharT std::num_get< _CharT, _InIter >::char_type
```

Public typedefs.

Definition at line 1958 of file locale\_facets.h.

#### 5.943.2.2 iter\_type

```
template<typename _CharT , typename _InIter >
typedef _InIter std::num_get< _CharT, _InIter >::iter_type
```

Public typedefs.

Definition at line 1959 of file locale\_facets.h.

### 5.943.3 Constructor & Destructor Documentation

#### 5.943.3.1 num\_get()

```
template<typename _CharT , typename _InIter >
std::num_get< _CharT, _InIter >::num_get (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1973 of file locale\_facets.h.

#### 5.943.3.2 ~num\_get()

```
template<typename _CharT , typename _InIter >
virtual std::num_get< _CharT, _InIter >::~num_get ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 2145 of file locale\_facets.h.

#### 5.943.4 Member Function Documentation

##### 5.943.4.1 `do_get()` [1/11]

```
template<typename _CharT, typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    bool & __v ) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

##### See also

`get()` for more details.

##### Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

##### Returns

Iterator after reading.

Definition at line 595 of file `locale_facets.tcc`.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

Referenced by `std::num_get< _CharT, _InIter >::get()`.

#### 5.943.4.2 do\_get() [2/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long & __v ) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

<i>__beg</i>	Start of input stream.
<i>__end</i>	End of input stream.
<i>__io</i>	Source of locale and flags.
<i>__err</i>	Error flags to set.
<i>__v</i>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2215 of file locale\_facets.h.

#### 5.943.4.3 do\_get() [3/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned short & __v ) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.



**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2220 of file `locale_facets.h`.

**5.943.4.4 do\_get()** [4/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned int & __v ) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2225 of file `locale_facets.h`.

## 5.943.4.5 do\_get() [5/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned long & __v ) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2230 of file locale\_facets.h.

## 5.943.4.6 do\_get() [6/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long long & __v ) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2236 of file `locale_facets.h`.

**5.943.4.7 do\_get()** [7/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned long long & __v ) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2241 of file `locale_facets.h`.

#### 5.943.4.8 do\_get() [8/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    float & __v ) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 691 of file `locale_facets.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::ios_base::eofbit`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

#### 5.943.4.9 do\_get() [9/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    double & __v ) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 706 of file `locale_facets.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`.

**5.943.4.10 do\_get()** [10/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get<_CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __v ) const    [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 738 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

**5.943.4.11 do\_get()** [11/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    void *& __v ) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

<i>__beg</i>	Start of input stream.
<i>__end</i>	End of input stream.
<i>__io</i>	Source of locale and flags.
<i>__err</i>	Error flags to set.
<i>__v</i>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 753 of file locale\_facets.tcc.

References std::ios\_base::basefield, std::ios\_base::flags(), and std::ios\_base::hex.

**5.943.4.12** `get()` [1/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    bool & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 1999 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

**5.943.4.13** `get()` [2/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2036 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

#### 5.943.4.14 get() [3/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned short & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.



**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2041 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

**5.943.4.15 `get()`** [4/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned int & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2046 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

**5.943.4.16 get()** [5/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    unsigned long & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2051 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

**5.943.4.17** `get()` [6/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long long & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2057 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

**5.943.4.18** `get()` [7/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
```

```

ios_base & __io,
ios_base::iostate & __err,
unsigned long long & __v ) const [inline]

```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2062 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

#### 5.943.4.19 get() [8/11]

```

template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    float & __v ) const [inline]

```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2096 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

#### 5.943.4.20 `get()` [9/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get<_CharT, _InIter>::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    double & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after reading.

Definition at line 2101 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

5.943.4.21 `get()` [10/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    long double & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2106 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

**5.943.4.22 get()** [11/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
    iter_type __in,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    void *& __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf p` specifier.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

**Parameters**

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2139 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

## 5.943.5 Member Data Documentation

## 5.943.5.1 id

```
template<typename _CharT , typename _InIter >  
locale::id std::num_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

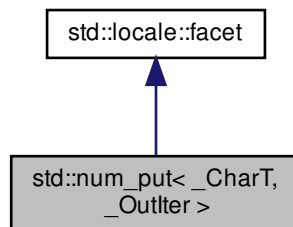
Definition at line 1963 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

## 5.944 std::num\_put&lt; \_CharT, \_OutIter &gt; Class Template Reference

Inheritance diagram for std::num\_put< \_CharT, \_OutIter >:



## Public Types

- typedef \_CharT [char\\_type](#)
- typedef \_OutIter [iter\\_type](#)



## Public Member Functions

- `num_put` (size\_t \_\_refs=0)
- `template<typename _ValueT >`  
`_Outlter _M_insert_float` (\_Outlter \_\_s, ios\_base & \_\_io, \_CharT \_\_fill, char \_\_mod, \_ValueT \_\_v) const
- `template<typename _ValueT >`  
`_Outlter _M_insert_int` (\_Outlter \_\_s, ios\_base & \_\_io, \_CharT \_\_fill, \_ValueT \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, bool \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const void \*\_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long long \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long long \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, double \_\_v) const
- `iter_type put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long double \_\_v) const

## Static Public Attributes

- static `locale::id` id

## Protected Member Functions

- virtual `~num_put` ()
- `void _M_group_float` (const char \*\_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, const char\_type \*\_\_p, char\_type \*\_\_new, char\_type \*\_\_cs, int & \_\_len) const
- `void _M_group_int` (const char \*\_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, ios\_base & \_\_io, char\_type \*\_\_new, char\_type \*\_\_cs, int & \_\_len) const
- `template<typename _ValueT >`  
`iter_type _M_insert_float` (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, char \_\_mod, \_ValueT \_\_v) const
- `template<typename _ValueT >`  
`iter_type _M_insert_int` (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, \_ValueT \_\_v) const
- `void _M_pad` (char\_type \_\_fill, streamsize \_\_w, ios\_base & \_\_io, char\_type \*\_\_new, const char\_type \*\_\_cs, int & \_\_len) const
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, bool \_\_v) const
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long \_\_v) const
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long \_\_v) const
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long long \_\_v) const
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long long \_\_v) const
- virtual `iter_type do_put` (iter\_type, ios\_base & \_\_io, char\_type, double) const
- virtual `iter_type do_put` (iter\_type, ios\_base & \_\_io, char\_type, long double) const
- virtual `iter_type do_put` (iter\_type, ios\_base & \_\_io, char\_type, const void \*) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 5.944.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::num_put<_CharT, _OutIter>
```

Primary class template num\_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

The num\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_put facet.

Definition at line 2293 of file locale\_facets.h.

#### 5.944.2 Member Typedef Documentation

##### 5.944.2.1 char\_type

```
template<typename _CharT , typename _OutIter >
typedef _CharT std::num_put<_CharT, _OutIter>::char_type
```

Public typedefs.

Definition at line 2299 of file locale\_facets.h.

##### 5.944.2.2 iter\_type

```
template<typename _CharT , typename _OutIter >
typedef _OutIter std::num_put<_CharT, _OutIter>::iter_type
```

Public typedefs.

Definition at line 2300 of file locale\_facets.h.

### 5.944.3 Constructor & Destructor Documentation

#### 5.944.3.1 num\_put()

```
template<typename _CharT , typename _OutIter >  
std::num_put< _CharT, _OutIter >::num_put (   
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 2314 of file locale\_facets.h.

## 5.944.3.2 ~num\_put()

```
template<typename _CharT , typename _OutIter >
virtual std::num_put<_CharT, _OutIter>::~~num_put ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 2493 of file locale\_facets.h.

## 5.944.4 Member Function Documentation

## 5.944.4.1 do\_put() [1/8]

```
template<typename _CharT , typename _OutIter >
_OutIter std::num_put<_CharT, _OutIter>::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    bool __v ) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 1106 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::adjustfield`, `std::ios_base::boolalpha`, `std::ios_base::flags()`, `std::ios_base::left`, and `std::ios_base::width()`.

Referenced by `std::num_put<_CharT, _OutIter >::put()`.

**5.944.4.2 do\_put()** [2/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long __v ) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2513 of file locale\_facets.h.

**5.944.4.3 do\_put()** [3/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
```

```

iter_type __s,
ios_base & __io,
char_type __fill,
unsigned long __v ) const [inline], [protected], [virtual]

```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2517 of file locale\_facets.h.

#### 5.944.4.4 do\_put() [4/8]

```

template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long long __v ) const [inline], [protected], [virtual]

```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2523 of file locale\_facets.h.

**5.944.4.5 do\_put()** [5/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    unsigned long long __v ) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2528 of file locale\_facets.h.

**5.944.4.6 do\_put()** [6/8]

```
template<typename _CharT , typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    double __v ) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 1158 of file locale\_facets.tcc.

**5.944.4.7 do\_put()** [7/8]

```
template<typename _CharT , typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long double __v ) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 1172 of file locale\_facets.tcc.



**5.944.4.8 do\_put()** [8/8]

```
template<typename _CharT , typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const void * __v ) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 1179 of file locale\_facets.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, `std::ios_base::showbase`, and `std::ios_base::uppercase`.

**5.944.4.9 put()** [1/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    bool __v ) const [inline]
```

Numeric formatting.

Formats the boolean `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Otherwise formats `v` as an int.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2332 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 5.944.4.10 put() [2/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long __v ) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2374 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

5.944.4.11 `put()` [3/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    unsigned long __v ) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2378 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 5.944.4.12 put() [4/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long long __v ) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2384 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

5.944.4.13 `put()` [5/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    unsigned long long __v ) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2388 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 5.944.4.14 put() [6/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    double __v ) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf *f* specifier. Else if equal to `ios_base::scientific`, formats like *e* or *E* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2437 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

5.944.4.15 `put()` [7/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    long double __v ) const [inline]
```

Numeric formatting.

Formats the floating point value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf `f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

## Returns

Iterator after writing.

Definition at line 2441 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 5.944.4.16 put() [8/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const void * __v ) const [inline]
```

Numeric formatting.

Formats the pointer value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats `v` as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

## Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.



**Returns**

Iterator after writing.

Definition at line 2462 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

**5.944.5 Member Data Documentation****5.944.5.1 id**

```
template<typename _CharT , typename _OutIter >
locale::id std::num_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

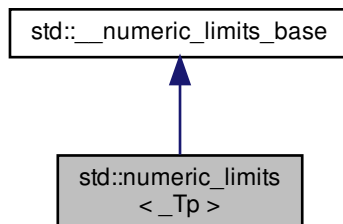
Definition at line 2304 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

**5.945 std::numeric\_limits<\_Tp> Struct Template Reference**

Inheritance diagram for `std::numeric_limits<_Tp>`:



## Static Public Member Functions

- static constexpr `_Tp` `denorm_min` () noexcept
- static constexpr `_Tp` `epsilon` () noexcept
- static constexpr `_Tp` `infinity` () noexcept
- static constexpr `_Tp` `lowest` () noexcept
- static constexpr `_Tp` `max` () noexcept
- static constexpr `_Tp` `min` () noexcept
- static constexpr `_Tp` `quiet_NaN` () noexcept
- static constexpr `_Tp` `round_error` () noexcept
- static constexpr `_Tp` `signaling_NaN` () noexcept

## Static Public Attributes

- static constexpr int `digits`
- static constexpr int `digits10`
- static constexpr `float_denorm_style` `has_denorm`
- static constexpr bool `has_denorm_loss`
- static constexpr bool `has_infinity`
- static constexpr bool `has_quiet_NaN`
- static constexpr bool `has_signaling_NaN`
- static constexpr bool `is_bounded`
- static constexpr bool `is_exact`
- static constexpr bool `is_iec559`
- static constexpr bool `is_integer`
- static constexpr bool `is_modulo`
- static constexpr bool `is_signed`
- static constexpr bool `is_specialized`
- static constexpr int `max_digits10`
- static constexpr int `max_exponent`
- static constexpr int `max_exponent10`
- static constexpr int `min_exponent`
- static constexpr int `min_exponent10`
- static constexpr int `radix`
- static constexpr `float_round_style` `round_style`
- static constexpr bool `tinyness_before`
- static constexpr bool `traps`

## 5.945.1 Detailed Description

```
template<typename _Tp>
struct std::numeric_limits<_Tp>
```

Properties of fundamental types.

This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

Definition at line 312 of file `limits`.

## 5.945.2 Member Function Documentation

### 5.945.2.1 `denorm_min()`

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::denorm_min ( ) [inline], [static], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

Definition at line 357 of file `limits`.

### 5.945.2.2 `epsilon()`

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::epsilon ( ) [inline], [static], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 333 of file `limits`.

Referenced by `std::generate_canonical()`, `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

### 5.945.2.3 `infinity()`

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::infinity ( ) [inline], [static], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

Definition at line 341 of file `limits`.

### 5.945.2.4 `lowest()`

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::lowest ( ) [inline], [static], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 327 of file `limits`.

Referenced by `std::normal_distribution< result_type >::min()`, `std::cauchy_distribution< _RealType >::min()`, `std::student_t_distribution< _RealType >::min()`, and `std::extreme_value_distribution< _RealType >::min()`.

### 5.945.2.5 max()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::max ( ) [inline], [static], [noexcept]
```

The maximum finite value.

Definition at line 321 of file limits.

Referenced by std::normal\_distribution< result\_type >::max(), std::lognormal\_distribution< \_RealType >::max(), std::gamma\_distribution< result\_type >::max(), std::chi\_squared\_distribution< \_RealType >::max(), std::cauchy\_distribution< \_RealType >::max(), std::fisher\_f\_distribution< \_RealType >::max(), std::student\_t\_distribution< \_RealType >::max(), std::bernoulli\_distribution::max(), std::geometric\_distribution< \_IntType >::max(), std::negative\_binomial\_distribution< \_IntType >::max(), std::poisson\_distribution< \_IntType >::max(), std::exponential\_distribution< \_RealType >::max(), std::weibull\_distribution< \_RealType >::max(), std::extreme\_value\_distribution< \_RealType >::max(), std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >::operator>(), std::binomial\_distribution< \_IntType >::operator(), and std::poisson\_distribution< \_IntType >::operator().

### 5.945.2.6 min()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::min ( ) [inline], [static], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 317 of file limits.

Referenced by std::bernoulli\_distribution::min(), and std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >::operator().

### 5.945.2.7 quiet\_NaN()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN ( ) [inline], [static], [noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

Definition at line 346 of file limits.

### 5.945.2.8 round\_error()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::round_error ( ) [inline], [static], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

Definition at line 337 of file limits.

#### 5.945.2.9 signaling\_NaN()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::signaling_NaN ( ) [inline], [static], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

Definition at line 351 of file limits.

### 5.945.3 Member Data Documentation

#### 5.945.3.1 digits

```
constexpr int std::__numeric_limits_base::digits [static], [inherited]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 211 of file limits.

#### 5.945.3.2 digits10

```
constexpr int std::__numeric_limits_base::digits10 [static], [inherited]
```

The number of base 10 digits that can be represented without change.

Definition at line 214 of file limits.

#### 5.945.3.3 has\_denorm

```
constexpr float_denorm_style std::__numeric_limits_base::has_denorm [static], [inherited]
```

See `std::float_denorm_style` for more information.

Definition at line 266 of file limits.

#### 5.945.3.4 has\_denorm\_loss

```
constexpr bool std::__numeric_limits_base::has_denorm_loss [static], [inherited]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 270 of file limits.

#### 5.945.3.5 has\_infinity

```
constexpr bool std::__numeric_limits_base::has_infinity [static], [inherited]
```

True if the type has a representation for positive infinity.

Definition at line 255 of file limits.

#### 5.945.3.6 has\_quiet\_NaN

```
constexpr bool std::__numeric_limits_base::has_quiet_NaN [static], [inherited]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 259 of file limits.

#### 5.945.3.7 has\_signaling\_NaN

```
constexpr bool std::__numeric_limits_base::has_signaling_NaN [static], [inherited]
```

True if the type has a representation for a signaling Not a Number.

Definition at line 263 of file limits.

#### 5.945.3.8 is\_bounded

```
constexpr bool std::__numeric_limits_base::is_bounded [static], [inherited]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 279 of file limits.

#### 5.945.3.9 is\_exact

```
constexpr bool std::__numeric_limits_base::is_exact [static], [inherited]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 231 of file limits.

#### 5.945.3.10 `is_iec559`

```
constexpr bool std::__numeric_limits_base::is_iec559 [static], [inherited]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 274 of file limits.

#### 5.945.3.11 `is_integer`

```
constexpr bool std::__numeric_limits_base::is_integer [static], [inherited]
```

True if the type is integer.

Definition at line 226 of file limits.

#### 5.945.3.12 `is_modulo`

```
constexpr bool std::__numeric_limits_base::is_modulo [static], [inherited]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max()]`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 288 of file limits.

#### 5.945.3.13 `is_signed`

```
constexpr bool std::__numeric_limits_base::is_signed [static], [inherited]
```

True if the type is signed.

Definition at line 223 of file limits.

#### 5.945.3.14 `is_specialized`

```
constexpr bool std::__numeric_limits_base::is_specialized [static], [inherited]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 206 of file limits.

#### 5.945.3.15 max\_digits10

```
constexpr int std::__numeric_limits_base::max_digits10 [static], [inherited]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 219 of file limits.

#### 5.945.3.16 max\_exponent

```
constexpr int std::__numeric_limits_base::max_exponent [static], [inherited]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 248 of file limits.

#### 5.945.3.17 max\_exponent10

```
constexpr int std::__numeric_limits_base::max_exponent10 [static], [inherited]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 252 of file limits.

#### 5.945.3.18 min\_exponent

```
constexpr int std::__numeric_limits_base::min_exponent [static], [inherited]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 239 of file limits.

#### 5.945.3.19 min\_exponent10

```
constexpr int std::__numeric_limits_base::min_exponent10 [static], [inherited]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 243 of file limits.



#### 5.945.3.20 radix

```
constexpr int std::__numeric_limits_base::radix [static], [inherited]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 235 of file limits.

#### 5.945.3.21 round\_style

```
constexpr float_round_style std::__numeric_limits_base::round_style [static], [inherited]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 299 of file limits.

#### 5.945.3.22 tinyness\_before

```
constexpr bool std::__numeric_limits_base::tinyness_before [static], [inherited]
```

True if tininess is detected before rounding. (see IEC 559)

Definition at line 294 of file limits.

#### 5.945.3.23 traps

```
constexpr bool std::__numeric_limits_base::traps [static], [inherited]
```

True if trapping is implemented for this type.

Definition at line 291 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.946 `std::numeric_limits< bool >` Struct Template Reference

### Static Public Member Functions

- static constexpr bool **denorm\_min** () noexcept
- static constexpr bool **epsilon** () noexcept
- static constexpr bool **infinity** () noexcept
- static constexpr bool **lowest** () noexcept
- static constexpr bool **max** () noexcept
- static constexpr bool **min** () noexcept
- static constexpr bool **quiet\_NaN** () noexcept
- static constexpr bool **round\_error** () noexcept
- static constexpr bool **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.946.1 Detailed Description

```
template<>
struct std::numeric_limits< bool >
```

`numeric_limits<bool>` specialization.

Definition at line 383 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.947 std::numeric\_limits< char > Struct Template Reference

### Static Public Member Functions

- static constexpr char **denorm\_min** () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char **max** () noexcept
- static constexpr char **min** () noexcept
- static constexpr char **quiet\_NaN** () noexcept
- static constexpr char **round\_error** () noexcept
- static constexpr char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float **denorm\_style** **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr float **round\_style** **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.947.1 Detailed Description

```
template<>
struct std::numeric_limits< char >
```

numeric\_limits<char> specialization.

Definition at line 452 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.948 `std::numeric_limits< char16_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr `char16_t` **denorm\_min** () noexcept
- static constexpr `char16_t` **epsilon** () noexcept
- static constexpr `char16_t` **infinity** () noexcept
- static constexpr `char16_t` **lowest** () noexcept
- static constexpr `char16_t` **max** () noexcept
- static constexpr `char16_t` **min** () noexcept
- static constexpr `char16_t` **quiet\_NaN** () noexcept
- static constexpr `char16_t` **round\_error** () noexcept
- static constexpr `char16_t` **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr `float_denorm_style` **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr `float_round_style` **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.948.1 Detailed Description

```
template<>
struct std::numeric_limits< char16_t >
```

`numeric_limits<char16_t>` specialization.

Definition at line 731 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.949 `std::numeric_limits< char32_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr char32\_t **denorm\_min** () noexcept
- static constexpr char32\_t **epsilon** () noexcept
- static constexpr char32\_t **infinity** () noexcept
- static constexpr char32\_t **lowest** () noexcept
- static constexpr char32\_t **max** () noexcept
- static constexpr char32\_t **min** () noexcept
- static constexpr char32\_t **quiet\_NaN** () noexcept
- static constexpr char32\_t **round\_error** () noexcept
- static constexpr char32\_t **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float **denorm\_style** **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr float **round\_style** **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.949.1 Detailed Description

```
template<>
struct std::numeric_limits< char32_t >
```

`numeric_limits<char32_t>` specialization.

Definition at line 792 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.950 `std::numeric_limits< double >` Struct Template Reference

### Static Public Member Functions

- static constexpr double **denorm\_min** () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double **max** () noexcept
- static constexpr double **min** () noexcept
- static constexpr double **quiet\_NaN** () noexcept
- static constexpr double **round\_error** () noexcept
- static constexpr double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.950.1 Detailed Description

```
template<>
struct std::numeric_limits< double >
```

`numeric_limits<double>` specialization.

Definition at line 1669 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.951 `std::numeric_limits< float >` Struct Template Reference

### Static Public Member Functions

- static constexpr float **denorm\_min** () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float **max** () noexcept
- static constexpr float **min** () noexcept
- static constexpr float **quiet\_NaN** () noexcept
- static constexpr float **round\_error** () noexcept
- static constexpr float **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.951.1 Detailed Description

```
template<>
struct std::numeric_limits< float >
```

`numeric_limits<float>` specialization.

Definition at line 1594 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.952 `std::numeric_limits< int >` Struct Template Reference

### Static Public Member Functions

- static constexpr int **denorm\_min** () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int **max** () noexcept
- static constexpr int **min** () noexcept
- static constexpr int **quiet\_NaN** () noexcept
- static constexpr int **round\_error** () noexcept
- static constexpr int **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.952.1 Detailed Description

```
template<>
struct std::numeric_limits< int >
```

`numeric_limits<int>` specialization.

Definition at line 994 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.953 `std::numeric_limits< long >` Struct Template Reference

### Static Public Member Functions

- static constexpr long **denorm\_min** () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long **max** () noexcept
- static constexpr long **min** () noexcept
- static constexpr long **quiet\_NaN** () noexcept
- static constexpr long **round\_error** () noexcept
- static constexpr long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.953.1 Detailed Description

```
template<>
struct std::numeric_limits< long >
```

`numeric_limits<long>` specialization.

Definition at line 1133 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.954 `std::numeric_limits< long double >` Struct Template Reference

## Static Public Member Functions

- static constexpr long double **denorm\_min** () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double **min** () noexcept
- static constexpr long double **quiet\_NaN** () noexcept
- static constexpr long double **round\_error** () noexcept
- static constexpr long double **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 5.954.1 Detailed Description

```
template<>
struct std::numeric_limits< long double >
```

`numeric_limits<long double>` specialization.

Definition at line 1744 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.955 `std::numeric_limits< long long >` Struct Template Reference

### Static Public Member Functions

- static constexpr long long **denorm\_min** () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long **quiet\_NaN** () noexcept
- static constexpr long long **round\_error** () noexcept
- static constexpr long long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float **denorm\_style** **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr float **round\_style** **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.955.1 Detailed Description

```
template<>
struct std::numeric_limits< long long >
```

`numeric_limits<long long>` specialization.

Definition at line 1273 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.956 `std::numeric_limits< short >` Struct Template Reference

### Static Public Member Functions

- static constexpr short **denorm\_min** () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short **max** () noexcept
- static constexpr short **min** () noexcept
- static constexpr short **quiet\_NaN** () noexcept
- static constexpr short **round\_error** () noexcept
- static constexpr short **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.956.1 Detailed Description

```
template<>
struct std::numeric_limits< short >
```

`numeric_limits<short>` specialization.

Definition at line 854 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.957 `std::numeric_limits< signed char >` Struct Template Reference

### Static Public Member Functions

- static constexpr signed char **denorm\_min** () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char **quiet\_NaN** () noexcept
- static constexpr signed char **round\_error** () noexcept
- static constexpr signed char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.957.1 Detailed Description

`template<>`

`struct std::numeric_limits< signed char >`

`numeric_limits<signed char>` specialization.

Definition at line 519 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.958 `std::numeric_limits< unsigned char >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned char **denorm\_min** () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char **min** () noexcept
- static constexpr unsigned char **quiet\_NaN** () noexcept
- static constexpr unsigned char **round\_error** () noexcept
- static constexpr unsigned char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.958.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned char >
```

`numeric_limits<unsigned char>` specialization.

Definition at line 589 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.959 `std::numeric_limits< unsigned int >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned int **denorm\_min** () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int **quiet\_NaN** () noexcept
- static constexpr unsigned int **round\_error** () noexcept
- static constexpr unsigned int **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.959.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned int >
```

`numeric_limits<unsigned int>` specialization.

Definition at line 1061 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.960 `std::numeric_limits< unsigned long >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned long **denorm\_min** () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long **quiet\_NaN** () noexcept
- static constexpr unsigned long **round\_error** () noexcept
- static constexpr unsigned long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.960.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned long >
```

`numeric_limits<unsigned long>` specialization.

Definition at line 1200 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)



## 5.961 `std::numeric_limits< unsigned long long >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned long long **denorm\_min** () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long **quiet\_NaN** () noexcept
- static constexpr unsigned long long **round\_error** () noexcept
- static constexpr unsigned long long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.961.1 Detailed Description

`template<>`

`struct std::numeric_limits< unsigned long long >`

`numeric_limits<unsigned long long>` specialization.

Definition at line 1343 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.962 `std::numeric_limits< unsigned short >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned short **denorm\_min** () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short **min** () noexcept
- static constexpr unsigned short **quiet\_NaN** () noexcept
- static constexpr unsigned short **round\_error** () noexcept
- static constexpr unsigned short **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.962.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned short >
```

`numeric_limits<unsigned short>` specialization.

Definition at line 921 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.963 `std::numeric_limits< wchar_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr `wchar_t` **denorm\_min** () noexcept
- static constexpr `wchar_t` **epsilon** () noexcept
- static constexpr `wchar_t` **infinity** () noexcept
- static constexpr `wchar_t` **lowest** () noexcept
- static constexpr `wchar_t` **max** () noexcept
- static constexpr `wchar_t` **min** () noexcept
- static constexpr `wchar_t` **quiet\_NaN** () noexcept
- static constexpr `wchar_t` **round\_error** () noexcept
- static constexpr `wchar_t` **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr `float_denorm_style` **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr `float_round_style` **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.963.1 Detailed Description

```
template<>
struct std::numeric_limits< wchar_t >
```

`numeric_limits<wchar_t>` specialization.

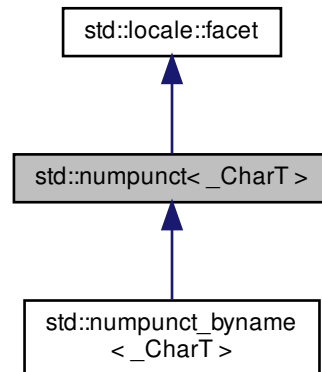
Definition at line 662 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.964 std::numpunct&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::numpunct<\_CharT>:



## Public Types

- typedef \_\_numpunct\_cache<\_CharT> **\_\_cache\_type**
- typedef \_CharT [char\\_type](#)
- typedef [basic\\_string](#)<\_CharT> [string\\_type](#)

## Public Member Functions

- [numpunct](#) (size\_t \_\_refs=0)
- [numpunct](#) (\_\_cache\_type \*\_\_cache, size\_t \_\_refs=0)
- [numpunct](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- [char\\_type](#) [decimal\\_point](#) () const
- [string\\_type](#) [falsename](#) () const
- [string](#) [grouping](#) () const
- [char\\_type](#) [thousands\\_sep](#) () const
- [string\\_type](#) [truename](#) () const

## Static Public Attributes

- static [locale::id](#) [id](#)

### Protected Member Functions

- virtual `~numpunct ()`
- void `_M_initialize_numpunct (__c_locale __cloc=0)`
- template<>  
void `_M_initialize_numpunct (__c_locale __cloc)`
- template<>  
void `_M_initialize_numpunct (__c_locale __cloc)`
- virtual `char_type do_decimal_point () const`
- virtual `string_type do_falsename () const`
- virtual `string do_grouping () const`
- virtual `char_type do_thousands_sep () const`
- virtual `string_type do_truename () const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- `__cache_type * _M_data`

#### 5.964.1 Detailed Description

```
template<typename _CharT>
class std::numpunct< _CharT >
```

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.

The `numpunct` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a `numpunct` facet.

Definition at line 1670 of file `locale_facets.h`.

#### 5.964.2 Member Typedef Documentation

### 5.964.2.1 char\_type

```
template<typename _CharT >
typedef _CharT std::numpunct< _CharT >::char_type
```

Public typedefs.

Definition at line 1676 of file locale\_facets.h.

### 5.964.2.2 string\_type

```
template<typename _CharT >
typedef basic_string<_CharT> std::numpunct< _CharT >::string_type
```

Public typedefs.

Definition at line 1677 of file locale\_facets.h.

## 5.964.3 Constructor & Destructor Documentation

### 5.964.3.1 numpunct() [1/3]

```
template<typename _CharT >
std::numpunct< _CharT >::numpunct (
    size_t __refs = 0 ) [inline], [explicit]
```

Numpunct constructor.

Parameters

<code>__refs</code>	Refcount to pass to the base class.
---------------------	-------------------------------------

Definition at line 1694 of file locale\_facets.h.

### 5.964.3.2 numpunct() [2/3]

```
template<typename _CharT >
std::numpunct< _CharT >::numpunct (
    __cache_type * __cache,
    size_t __refs = 0 ) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

**Parameters**

<code>__cache</code>	<code>__numpunct_cache</code> object.
<code>__refs</code>	Refcount to pass to the base class.

Definition at line 1708 of file `locale_facets.h`.

**5.964.3.3 numpunct()** [3/3]

```
template<typename _CharT >
std::numpunct< _CharT >::numpunct (
    __c_locale __cloc,
    size_t __refs = 0 ) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

<code>__cloc</code>	The C locale.
<code>__refs</code>	Refcount to pass to the base class.

Definition at line 1722 of file `locale_facets.h`.

**5.964.3.4 ~numpunct()**

```
template<typename _CharT >
virtual std::numpunct< _CharT >::~numpunct ( ) [protected], [virtual]
```

Destructor.

**5.964.4 Member Function Documentation****5.964.4.1 decimal\_point()**

```
template<typename _CharT >
char_type std::numpunct< _CharT >::decimal_point ( ) const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `numpunct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1736 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_decimal\_point().

**5.964.4.2 do\_decimal\_point()**

```
template<typename _CharT >
virtual char_type std::num_punct<_CharT>::do_decimal_point ( ) const [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1823 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::decimal\_point().

**5.964.4.3 do\_falsename()**

```
template<typename _CharT >
virtual string_type std::num_punct<_CharT>::do_falsename ( ) const [inline], [protected], [virtual]
```

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1874 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::falsename().



#### 5.964.4.4 do\_grouping()

```
template<typename _CharT >
virtual string std::num_punct< _CharT >::do_grouping ( ) const [inline], [protected], [virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1848 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::grouping().

#### 5.964.4.5 do\_thousands\_sep()

```
template<typename _CharT >
virtual char_type std::num_punct< _CharT >::do_thousands_sep ( ) const [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a thousands separator.

Definition at line 1835 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::thousands\_sep().

## 5.964.4.6 do\_truename()

```
template<typename _CharT >
virtual string_type std::num_punct<_CharT>::do_truename ( ) const [inline], [protected], [virtual]
```

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

string\_type representing printed form of true.

Definition at line 1861 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::truename().

## 5.964.4.7 falsename()

```
template<typename _CharT >
string_type std::num_punct<_CharT>::falsename ( ) const [inline]
```

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_falsename().

**Returns**

string\_type representing printed form of false.

Definition at line 1806 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_falsename().

#### 5.964.4.8 grouping()

```
template<typename _CharT >
string std::num_punct< _CharT >::grouping ( ) const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

##### Returns

string representing grouping specification.

Definition at line 1780 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_grouping().

#### 5.964.4.9 thousands\_sep()

```
template<typename _CharT >
char_type std::num_punct< _CharT >::thousands_sep ( ) const [inline]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

##### Returns

char\_type representing a thousands separator.

Definition at line 1749 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_thousands\_sep().

## 5.964.4.10 true\_name()

```
template<typename _CharT >
string_type std::numpunct<_CharT>::true_name ( ) const [inline]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling numpunct<char\_type>::do\_true\_name().

**Returns**

string\_type representing printed form of true.

Definition at line 1793 of file locale\_facets.h.

References std::numpunct<\_CharT>::do\_true\_name().

## 5.964.5 Member Data Documentation

## 5.964.5.1 id

```
template<typename _CharT >
locale::id std::numpunct<_CharT>::id [static]
```

Numpunct facet id.

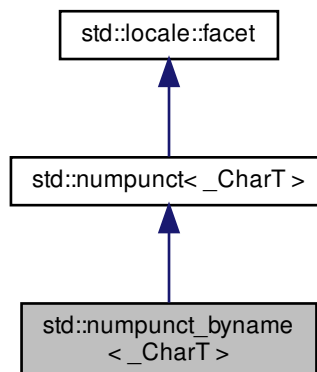
Definition at line 1686 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.965 std::numpunct\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::numpunct\_byname<\_CharT>:



### Public Types

- typedef \_\_numpunct\_cache<\_CharT> **\_\_cache\_type**
- typedef \_CharT **char\_type**
- typedef [basic\\_string](#)<\_CharT> **string\_type**

### Public Member Functions

- **numpunct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **numpunct\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- [char\\_type decimal\\_point](#) () const
- [string\\_type falsename](#) () const
- [string grouping](#) () const
- [char\\_type thousands\\_sep](#) () const
- [string\\_type truename](#) () const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc=0)
- template<>  
void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc)
- template<>  
void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc)
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_cache\_type \* **\_M\_data**

### 5.965.1 Detailed Description

```
template<typename _CharT>
class std::num_punct_byname<_CharT>
```

class num\_punct\_byname [22.2.3.2].

Definition at line 1903 of file locale\_facets.h.

### 5.965.2 Member Function Documentation

#### 5.965.2.1 decimal\_point()

```
template<typename _CharT >
char_type std::num_punct<_CharT>::decimal_point ( ) const [inline], [inherited]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `num_punct<char_type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1736 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_decimal_point()`.

#### 5.965.2.2 do\_decimal\_point()

```
template<typename _CharT >
virtual char_type std::num_punct<_CharT>::do_decimal_point ( ) const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1823 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

#### 5.965.2.3 do\_falsename()

```
template<typename _CharT >
virtual string_type std::num_punct< _CharT >::do_falsename ( ) const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool false.

Returns a string\_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

##### Returns

string\_type representing printed form of false.

Definition at line 1874 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::falsename().

#### 5.965.2.4 do\_grouping()

```
template<typename _CharT >
virtual string std::num_punct< _CharT >::do_grouping ( ) const [inline], [protected], [virtual],
[inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1848 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::grouping().

#### 5.965.2.5 do\_thousands\_sep()

```
template<typename _CharT >
virtual char_type std::num_punct<_CharT>::do_thousands_sep ( ) const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a thousands separator.

Definition at line 1835 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::thousands_sep()`.

#### 5.965.2.6 do\_truename()

```
template<typename _CharT >
virtual string_type std::num_punct<_CharT>::do_truename ( ) const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

##### Returns

`string_type` representing printed form of true.

Definition at line 1861 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::truename()`.

#### 5.965.2.7 falsename()

```
template<typename _CharT >
string_type std::num_punct<_CharT>::falsename ( ) const [inline], [inherited]
```

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

##### Returns

`string_type` representing printed form of false.

Definition at line 1806 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_falsename()`.



#### 5.965.2.8 grouping()

```
template<typename _CharT >
string std::num_punct< _CharT >::grouping ( ) const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

##### Returns

string representing grouping specification.

Definition at line 1780 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_grouping().

#### 5.965.2.9 thousands\_sep()

```
template<typename _CharT >
char_type std::num_punct< _CharT >::thousands_sep ( ) const [inline], [inherited]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

##### Returns

char\_type representing a thousands separator.

Definition at line 1749 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_thousands\_sep().

## 5.965.2.10 truename()

```
template<typename _CharT >
string_type std::numpunct< _CharT >::truename ( ) const [inline], [inherited]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling numpunct<char\_type>::do\_truename().

**Returns**

string\_type representing printed form of true.

Definition at line 1793 of file locale\_facets.h.

References std::numpunct< \_CharT >::do\_truename().

## 5.965.3 Member Data Documentation

## 5.965.3.1 id

```
template<typename _CharT >
locale::id std::numpunct< _CharT >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 1686 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.966 std::once\_flag Struct Reference

**Public Member Functions**

- constexpr [once\\_flag](#) () noexcept=default
- [once\\_flag](#) (const [once\\_flag](#) &)=delete
- [once\\_flag](#) & [operator=](#) (const [once\\_flag](#) &)=delete

**Friends**

- template<typename \_Callable , typename... \_Args>  
void [call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)

### 5.966.1 Detailed Description

once\_flag

Definition at line 629 of file mutex.

### 5.966.2 Constructor & Destructor Documentation

#### 5.966.2.1 once\_flag() [1/2]

```
constexpr std::once_flag::once_flag ( ) [default], [noexcept]
```

Constructor.

#### 5.966.2.2 once\_flag() [2/2]

```
std::once_flag::once_flag (
    const once_flag & ) [delete]
```

Deleted copy constructor.

### 5.966.3 Member Function Documentation

#### 5.966.3.1 operator=()

```
once_flag& std::once_flag::operator= (
    const once_flag & ) [delete]
```

Deleted assignment operator.

### 5.966.4 Friends And Related Function Documentation

## 5.966.4.1 call\_once

```
template<typename _Callable , typename... _Args>
void call_once (
    once_flag & __once,
    _Callable && __f,
    _Args &&... __args ) [friend]
```

call\_once

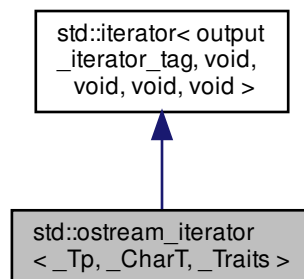
Definition at line 667 of file mutex.

The documentation for this struct was generated from the following file:

- [mutex](#)

## 5.967 std::ostream\_iterator&lt; \_Tp, \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::ostream\_iterator< \_Tp, \_CharT, \_Traits >:



## Public Types

- typedef void [difference\\_type](#)
  - typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
  - typedef void [pointer](#)
  - typedef void [reference](#)
  - typedef void [value\\_type](#)
- 
- typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
  - typedef [basic\\_ostream< \\_CharT, \\_Traits >](#) [ostream\\_type](#)

## Public Member Functions

- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s)
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const [\\_CharT](#) \*\_\_c)
- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj)
- [ostream\\_iterator](#) & **operator\*** ()
- [ostream\\_iterator](#) & **operator++** ()
- [ostream\\_iterator](#) & **operator++** (int)
- [ostream\\_iterator](#) & **operator=** (const [\\_Tp](#) &\_\_value)

### 5.967.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >
```

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

#### Template Parameters

<code>_Tp</code>	The type to write to the ostream.
<code>_CharT</code>	The ostream <code>char_type</code> .
<code>_Traits</code>	The ostream <code>char_traits</code> .

Definition at line 154 of file `stream_iterator.h`.

### 5.967.2 Member Typedef Documentation

#### 5.967.2.1 `char_type`

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _CharT std::ostream\_iterator< _Tp, _CharT, _Traits >::char\_type
```

Public typedef.

Definition at line 160 of file `stream_iterator.h`.

### 5.967.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type  
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

### 5.967.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >←  
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

### 5.967.2.4 ostream\_type

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>  
typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type
```

Public typedef.

Definition at line 162 of file stream\_iterator.h.

### 5.967.2.5 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

### 5.967.2.6 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

### 5.967.2.7 traits\_type

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type
```

Public typedef.

Definition at line 161 of file stream\_iterator.h.

### 5.967.2.8 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

## 5.967.3 Constructor & Destructor Documentation

### 5.967.3.1 ostream\_iterator() [1/3]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
    ostream_type & __s ) [inline]
```

Construct from an ostream.

Definition at line 171 of file stream\_iterator.h.



### 5.967.3.2 ostream\_iterator() [2/3]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
    ostream_type & __s,
    const _CharT * __c ) [inline]
```

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

## Parameters

<a href="#">_s</a>	Underlying ostream to write to.
<a href="#">_c</a>	CharT delimiter string to insert.

Definition at line 184 of file stream\_iterator.h.

## 5.967.3.3 ostream\_iterator() [3/3]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
    const ostream_iterator< _Tp, _CharT, _Traits > & __obj ) [inline]
```

Copy constructor.

Definition at line 188 of file stream\_iterator.h.

## 5.967.4 Member Function Documentation

## 5.967.4.1 operator=()

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
ostream_iterator& std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (
    const _Tp & __value ) [inline]
```

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.

Definition at line 194 of file stream\_iterator.h.

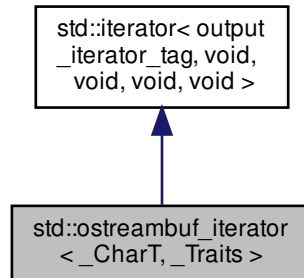
The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)



## 5.968 `std::ostreambuf_iterator< _CharT, _Traits >` Class Template Reference

Inheritance diagram for `std::ostreambuf_iterator< _CharT, _Traits >`:



### Public Types

- typedef void [difference\\_type](#)
  - typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
  - typedef void [pointer](#)
  - typedef void [reference](#)
  - typedef void [value\\_type](#)
- 
- typedef `_CharT` [char\\_type](#)
  - typedef `_Traits` [traits\\_type](#)
  - typedef [basic\\_ostreambuf\\_iterator< \\_CharT, \\_Traits >](#) [ostreambuf\\_type](#)
  - typedef [basic\\_ostream< \\_CharT, \\_Traits >](#) [ostream\\_type](#)

### Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostreambuf\\_iterator](#) ([ostreambuf\\_type](#) \*\_\_s) noexcept
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const `_CharT` \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const noexcept
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator=](#) (`_CharT` \_\_c)

## Friends

- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type >` **copy**  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)`

### 5.968.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::ostreambuf_iterator< _CharT, _Traits >
```

Provides output iterator semantics for streambufs.

Definition at line 128 of file iosfwd.

### 5.968.2 Member Typedef Documentation

#### 5.968.2.1 char\_type

```
template<typename _CharT , typename _Traits >
typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

Definition at line 222 of file ostreambuf\_iterator.h.

#### 5.968.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file stl\_iterator\_base\_types.h.

#### 5.968.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >←
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

#### 5.968.2.4 ostream\_type

```
template<typename _CharT , typename _Traits >
typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type
```

Public typedefs.

Definition at line 225 of file streambuf\_iterator.h.

#### 5.968.2.5 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

#### 5.968.2.6 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

#### 5.968.2.7 streambuf\_type

```
template<typename _CharT , typename _Traits >
typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf_type
```

Public typedefs.

Definition at line 224 of file streambuf\_iterator.h.

#### 5.968.2.8 traits\_type

```
template<typename _CharT , typename _Traits >
typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type
```

Public typedefs.

Definition at line 223 of file streambuf\_iterator.h.

### 5.968.2.9 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

## 5.968.3 Constructor & Destructor Documentation

### 5.968.3.1 ostreambuf\_iterator() [1/2]

```
template<typename _CharT , typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
    ostream_type & __s ) [inline], [noexcept]
```

Construct output iterator from ostream.

Definition at line 240 of file ostreambuf\_iterator.h.

### 5.968.3.2 ostreambuf\_iterator() [2/2]

```
template<typename _CharT , typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
    streambuf_type * __s ) [inline], [noexcept]
```

Construct output iterator from streambuf.

Definition at line 244 of file streambuf\_iterator.h.

## 5.968.4 Member Function Documentation

### 5.968.4.1 failed()

```
template<typename _CharT , typename _Traits >
bool std::ostreambuf_iterator< _CharT, _Traits >::failed ( ) const [inline], [noexcept]
```

Return true if previous operator=() failed.

Definition at line 274 of file streambuf\_iterator.h.

#### 5.968.4.2 operator\*()

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator* ( ) [inline]
```

Return \*this.

Definition at line 259 of file ostreambuf\_iterator.h.

#### 5.968.4.3 operator++() [1/2]

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++ (
    int ) [inline]
```

Return \*this.

Definition at line 264 of file ostreambuf\_iterator.h.

#### 5.968.4.4 operator++() [2/2]

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++ ( ) [inline]
```

Return \*this.

Definition at line 269 of file ostreambuf\_iterator.h.

#### 5.968.4.5 operator=()

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator= (
    _CharT __c ) [inline]
```

Write character to ostreambuf. Calls ostreambuf.sputc().

Definition at line 249 of file ostreambuf\_iterator.h.

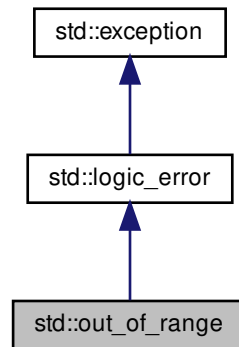
References std::basic\_ostreambuf< \_CharT, \_Traits >::sputc().

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [ostreambuf\\_iterator.h](#)

## 5.969 std::out\_of\_range Class Reference

Inheritance diagram for std::out\_of\_range:



### Public Member Functions

- **out\_of\_range** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **out\_of\_range** (const char \*) \_GLIBCXX\_TXN\_SAFE
- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

### 5.969.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in `basic_string`).

Definition at line 182 of file `stdexcept`.

### 5.969.2 Member Function Documentation

#### 5.969.2.1 [what\(\)](#)

```
virtual const char* std::logic_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.970 `std::output_iterator_tag` Struct Reference

### 5.970.1 Detailed Description

Marking output iterators.

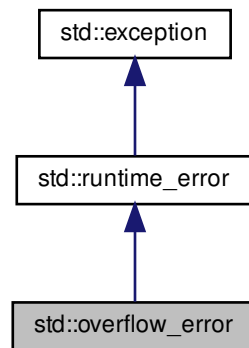
Definition at line 92 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.971 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- **`overflow_error`** (const [string](#) &\_\_arg) `_GLIBCXX_TXN_SAFE`
- **`overflow_error`** (const char \*) `_GLIBCXX_TXN_SAFE`
- virtual const char \* [what](#) () const `_GLIBCXX_TXN_SAFE_DYN noexcept`

### 5.971.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 241 of file `stdexcept`.

## 5.971.2 Member Function Documentation

5.971.2.1 `what()`

```
virtual const char* std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.972 `std::owner_less< _Tp >` Struct Template Reference

## 5.972.1 Detailed Description

```
template<typename _Tp = void>  
struct std::owner_less< _Tp >
```

Primary template `owner_less`.

Definition at line 616 of file `bits/shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

5.973 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::_Sp_owner_less< _Tp, _Tp1 >`.

## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept
- `bool` **operator()** (const `_Tp` &\_\_lhs, const `_Tp1` &\_\_rhs) const noexcept
- `bool` **operator()** (const `_Tp1` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept



### 5.973.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of owner\_less for shared\_ptr.

Definition at line 625 of file bits/shared\_ptr.h.

### 5.973.2 Member Typedef Documentation

#### 5.973.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 5.973.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 5.973.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

### 5.974 std::owner\_less< void > Struct Template Reference

Inherits std::\_Sp\_owner\_less< \_Tp, \_Tp1 >.

## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- `bool` **operator()** (`const _Tp &__lhs, const _Tp &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp &__lhs, const _Tp1 &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp1 &__lhs, const _Tp &__rhs`) `const noexcept`

### 5.974.1 Detailed Description

```
template<>
struct std::owner_less< void >
```

Void specialization of `owner_less`.

Definition at line 620 of file `bits/shared_ptr.h`.

### 5.974.2 Member Typedef Documentation

#### 5.974.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 5.974.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

### 5.974.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.975 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_Sp\_owner\_less< \_Tp, \_Tp1 >.

### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 5.975.1 Detailed Description

```
template<typename _Tp>  
struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of owner\_less for weak\_ptr.

Definition at line 631 of file bits/shared\_ptr.h.

### 5.975.2 Member Typedef Documentation

## 5.975.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

## 5.975.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 5.975.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 5.976 std::packaged\_task&lt; \_Res(\_ArgTypes...)&gt; Class Template Reference

## Public Member Functions

- template<typename \_Allocator >  
**packaged\_task** (allocator\_arg\_t, const \_Allocator &\_\_a) noexcept
- template<typename \_Fn, typename = typename \_\_constrain\_pkgdtask<packaged\_task, \_Fn>::\_\_type>  
**packaged\_task** (\_Fn &&\_\_fn)
- template<typename \_Fn, typename \_Alloc, typename = typename \_\_constrain\_pkgdtask<packaged\_task, \_Fn>::\_\_type>  
**packaged\_task** (allocator\_arg\_t, const \_Alloc &\_\_a, \_Fn &&\_\_fn)
- **packaged\_task** (const packaged\_task &)=delete
- template<typename \_Allocator >  
**packaged\_task** (allocator\_arg\_t, const \_Allocator &, const packaged\_task &)=delete
- **packaged\_task** (packaged\_task &&\_\_other) noexcept
- template<typename \_Allocator >  
**packaged\_task** (allocator\_arg\_t, const \_Allocator &, packaged\_task &&\_\_other) noexcept
- **future**< \_Res > **get\_future** ()
- void **make\_ready\_at\_thread\_exit** (\_ArgTypes... \_\_args)
- void **operator()** (\_ArgTypes... \_\_args)
- packaged\_task & **operator=** (const packaged\_task &)=delete
- packaged\_task & **operator=** (packaged\_task &&\_\_other) noexcept
- void **reset** ()
- void **swap** (packaged\_task &\_\_other) noexcept
- bool **valid** () const noexcept

### 5.976.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
class std::packaged_task< _Res(_ArgTypes...)>
```

packaged\_task

Definition at line 1476 of file future.

The documentation for this class was generated from the following file:

- [future](#)

### 5.977 std::pair< \_T1, \_T2 > Struct Template Reference

Inherits std::\_\_pair\_base< \_U1, \_U2 >.

#### Public Types

- template<typename \_U1, typename \_U2 >  
using [\\_PCCFP](#) = [\\_PCC](#)<[is\\_same](#)< \_T1, \_U1 >::value||[is\\_same](#)< \_T2, \_U2 >::value, \_T1, \_T2 >
- using [\\_PCCP](#) = [\\_PCC](#)< true, \_T1, \_T2 >
- typedef \_T1 **first\_type**
- typedef \_T2 [second\\_type](#)

#### Public Member Functions

- template<typename \_U1 = \_T1, typename \_U2 = \_T2, typename enable\_if< \_\_and< \_\_is\_implicitly\_default\_constructible< \_U1 >, \_\_is\_implicitly\_default\_constructible< \_U2 >> ::value, bool >::type = true>  
constexpr [pair](#) ()
- template<typename \_U1 = \_T1, typename \_U2 = \_T2, typename enable\_if< \_PCCP::template \_ConstructiblePair< \_U1, \_U2 >() &&\_\_PCCP::template \_ImplicitlyConvertiblePair< \_U1, \_U2 >(), bool >::type = true>  
constexpr **pair** (const \_T1 &\_\_a, const \_T2 &\_\_b)
- template<typename \_U1 = \_T1, typename \_U2 = \_T2, typename enable\_if< \_PCCP::template \_ConstructiblePair< \_U1, \_U2 >() &&!\_\_PCCP::template \_ImplicitlyConvertiblePair< \_U1, \_U2 >(), bool >::type = false>  
constexpr **pair** (const \_T1 &\_\_a, const \_T2 &\_\_b)
- template<typename \_U1, typename \_U2, typename enable\_if< \_PCCFP< \_U1, \_U2 >::template \_ConstructiblePair< \_U1, \_U2 >() &&\_PCCFP< \_U1, \_U2 >::template \_ImplicitlyConvertiblePair< \_U1, \_U2 >(), bool >::type = true>  
constexpr **pair** (const [pair](#)< \_U1, \_U2 > &\_\_p)
- template<typename \_U1, typename \_U2, typename enable\_if< \_PCCFP< \_U1, \_U2 >::template \_ConstructiblePair< \_U1, \_U2 >() &&!\_PCCFP< \_U1, \_U2 >::template \_ImplicitlyConvertiblePair< \_U1, \_U2 >(), bool >::type = false>  
constexpr **pair** (const [pair](#)< \_U1, \_U2 > &\_\_p)
- constexpr **pair** (const [pair](#) &)=default
- constexpr **pair** ([pair](#) &&)=default
- template<typename \_U1, typename enable\_if< \_PCCP::template \_MoveCopyPair< true, \_U1, \_T2 >(), bool >::type = true>  
constexpr **pair** (\_U1 &&\_\_x, const \_T2 &\_\_y)

- `template<typename _U1, typename enable_if<_PCCP::template _MoveCopyPair<false, _U1, _T2>(), bool>::type = false> constexpr pair ( _U1 &&__x, const _T2 &__y)`
- `template<typename _U2, typename enable_if<_PCCP::template _CopyMovePair<true, _T1, _U2>(), bool>::type = true> constexpr pair (const _T1 &__x, _U2 &&__y)`
- `template<typename _U2, typename enable_if<_PCCP::template _CopyMovePair<false, _T1, _U2>(), bool>::type = false> pair (const _T1 &__x, _U2 &&__y)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCP::template _MoveConstructiblePair<_U1, _U2>() &&_PCCP::template _ImplicitlyMoveConvertiblePair<_U1, _U2>(), bool>::type = true> constexpr pair ( _U1 &&__x, _U2 &&__y)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCP::template _MoveConstructiblePair<_U1, _U2>() &&!_PCCP::template _ImplicitlyMoveConvertiblePair<_U1, _U2>(), bool>::type = false> constexpr pair ( _U1 &&__x, _U2 &&__y)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCFP<_U1, _U2>::template _MoveConstructiblePair<_U1, _U2>() &&_PCCFP<_U1, _U2>::template _ImplicitlyMoveConvertiblePair<_U1, _U2>(), bool>::type = true> constexpr pair (pair<_U1, _U2> &&__p)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCFP<_U1, _U2>::template _MoveConstructiblePair<_U1, _U2>() &&_PCCFP<_U1, _U2>::template _ImplicitlyMoveConvertiblePair<_U1, _U2>(), bool>::type = false> constexpr pair (pair<_U1, _U2> &&__p)`
- `template<typename... _Args1, typename... _Args2> pair (piecewise_construct_t, tuple<_Args1...>, tuple<_Args2...>)`
- `pair & operator= (typename conditional<__and<is_copy_assignable<_T1>, is_copy_assignable<_T2>>::value, const pair &, const __nonesuch_no_braces &>::type __p)`
- `pair & operator= (typename conditional<__and<is_move_assignable<_T1>, is_move_assignable<_T2>>::value, pair &&, __nonesuch_no_braces &&>::type __p) noexcept(__and<is_nothrow_move_assignable<_T1>, is_nothrow_move_assignable<_T2>>::value)`
- `template<typename _U1, typename _U2> enable_if<__and<is_assignable<_T1 &, const _U1 &>, is_assignable<_T2 &, const _U2 &>>::value, pair &>::type operator= (const pair<_U1, _U2> &__p)`
- `template<typename _U1, typename _U2> enable_if<__and<is_assignable<_T1 &, _U1 &&>, is_assignable<_T2 &, _U2 &&>>::value, pair &>::type operator= (pair<_U1, _U2> &&__p)`
- `void swap (pair &__p) noexcept(__and<__is_nothrow_swappable<_T1>, __is_nothrow_swappable<_T2>>::value)`

#### Public Attributes

- `_T1` [first](#)
- `_T2` [second](#)

#### 5.977.1 Detailed Description

```
template<typename _T1, typename _T2>
struct std::pair<_T1, _T2>
```

Struct holding two objects of arbitrary type.

#### Template Parameters

<code>_T1</code>	Type of first object.
<code>_T2</code>	Type of second object.

Definition at line 208 of file stl\_pair.h.

## 5.977.2 Member Typedef Documentation

### 5.977.2.1 `_PCCFP`

```
template<typename _T1, typename _T2>
template<typename _U1 , typename _U2 >
using std::pair< _T1, _T2 >::_PCCFP = _PCC<!is_same<_T1, _U1>::value || !is_same<_T2, _U2>↵
::value, _T1, _T2>
```

There is also a templated copy ctor for the `pair` class itself.

Definition at line 283 of file stl\_pair.h.

### 5.977.2.2 `_PCCP`

```
template<typename _T1, typename _T2>
using std::pair< _T1, _T2 >::_PCCP = _PCC<true, _T1, _T2>
```

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 252 of file stl\_pair.h.

### 5.977.2.3 `second_type`

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

`first_type` is the first bound type

Definition at line 212 of file stl\_pair.h.

## 5.977.3 Constructor & Destructor Documentation

### 5.977.3.1 pair()

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and< __is_implicitly_↵
default_constructible< _U1 >, __is_implicitly_default_constructible< _U2 >> ::value, bool >↵
::type = true>
constexpr std::pair< _T1, _T2 >::pair ( ) [inline]
```

`second` is a copy of the second object

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 229 of file `stl_pair.h`.

### 5.977.4 Member Data Documentation

#### 5.977.4.1 first

```
template<typename _T1, typename _T2>
\_T1 std::pair< _T1, _T2 >::first
```

`second_type` is the second bound type

Definition at line 214 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `std::__sample()`, `__gnu_debug::__valid_range_aux()`, `std::set<_Key, _Compare, _Alloc >::insert()`, `std::operator<()`, `std::operator==()`, and `std::regex_replace()`.

#### 5.977.4.2 second

```
template<typename _T1, typename _T2>
\_T2 std::pair< _T1, _T2 >::second
```

`first` is a copy of the first object

Definition at line 215 of file `stl_pair.h`.

Referenced by `std::__sample()`, `__gnu_debug::__valid_range_aux()`, `std::set<_Key, _Compare, _Alloc >::insert()`, `std::operator<()`, `std::operator==()`, and `std::regex_replace()`.

The documentation for this struct was generated from the following files:

- [stl\\_pair.h](#)
- [tuple](#)



## 5.978 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**`piecewise_constant_distribution`** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**`piecewise_constant_distribution`** ([initializer\\_list](#)< `_RealType` > `__bl, _Func __fw`)
- `template<typename _Func >`  
**`piecewise_constant_distribution`** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **`piecewise_constant_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` ([result\\_type](#) \*`__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `std::vector< double > densities` () const
- `std::vector< _RealType > intervals` () const
- [result\\_type](#) `max` () const
- [result\\_type](#) `min` () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) `operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type](#) `param` () const
- `void param` (`const param\_type &__param`)
- `void reset` ()

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<` (`std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_constant_distribution< _RealType1 > &__x`)
- `bool operator==` (`const piecewise\_constant\_distribution &__d1, const piecewise\_constant\_distribution &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream< _CharT, _Traits > &__is, std::piecewise_constant_distribution< _RealType1 > &__x`)

### 5.978.1 Detailed Description

```
template<typename _RealType = double>
class std::piecewise_constant_distribution< _RealType >
```

A piecewise\_constant\_distribution random number distribution.

The formula for the piecewise constant probability mass function is

Definition at line 5406 of file random.h.

### 5.978.2 Member Typedef Documentation

#### 5.978.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 5409 of file random.h.

### 5.978.3 Member Function Documentation

#### 5.978.3.1 densities()

```
template<typename _RealType = double>
std::vector<double> std::piecewise_constant_distribution< _RealType >::densities ( ) const [inline]
```

Returns a vector of the probability densities.

Definition at line 5532 of file random.h.

References std::vector< \_Tp, \_Alloc >::empty().

### 5.978.3.2 intervals()

```
template<typename _RealType = double>
std::vector<_RealType> std::piecewise_constant_distribution< _RealType >::intervals ( ) const
[inline]
```

Returns a vector of the intervals.

Definition at line 5516 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

### 5.978.3.3 max()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5567 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`, and `std::vector< _Tp, _Alloc >::empty()`.

### 5.978.3.4 min()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5557 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::front()`.

### 5.978.3.5 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::piecewise_constant_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 5578 of file random.h.

**5.978.3.6 param()** [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_constant_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5542 of file random.h.

**5.978.3.7 param()** [2/2]

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5550 of file random.h.

**5.978.3.8 reset()**

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 5509 of file random.h.

**5.978.4 Friends And Related Function Documentation****5.978.4.1 operator<<**

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::piecewise_constant_distribution< _RealType1 > & __x ) [friend]
```

Inserts a piecewise\_constant\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A <code>piecewise_constant_distribution</code> random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**5.978.4.2 `operator==`**

```
template<typename _RealType = double>
bool operator== (
    const piecewise\_constant\_distribution< _RealType > & __d1,
    const piecewise\_constant\_distribution< _RealType > & __d2 ) [friend]
```

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5613 of file `random.h`.

**5.978.4.3 `operator>>`**

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic\_istream<_CharT, _Traits>& operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::piecewise\_constant\_distribution< _RealType1 > & __x ) [friend]
```

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>piecewise_constant_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.979 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `piecewise_constant_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**param\_type** (initializer\_list<\_RealType> \_\_bi, \_Func \_\_fw)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **param\_type** (const `param_type` &)=default
- `std::vector<double>` **densities** () const
- `std::vector<_RealType>` **intervals** () const
- `param_type` & **operator=** (const `param_type` &)=default

## Friends

- bool **operator!=** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- class `piecewise_constant_distribution<_RealType>`

## 5.979.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_constant_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 5416 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

5.980 `std::piecewise_construct_t` Struct Reference

## 5.980.1 Detailed Description

```
piecewise_construct_t
```

Definition at line 76 of file `stl_pair.h`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.981 `std::piecewise_linear_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**`piecewise_linear_distribution`** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**`piecewise_linear_distribution`** (`initializer_list<_RealType> __bl, _Func __fw`)
- `template<typename _Func >`  
**`piecewise_linear_distribution`** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **`piecewise_linear_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` (`result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `std::vector< double > densities` () const
- `std::vector<_RealType> intervals` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `param_type param` () const
- `void param` (`const param\_type &__param`)
- `void reset` ()

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits> &__os, const std::piecewise_linear_distribution<_RealType1> &__x`)
- `bool operator==` (`const piecewise\_linear\_distribution &__d1, const piecewise\_linear\_distribution &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream<_CharT, _Traits> & operator>>` (`std::basic_istream<_CharT, _Traits> &__is, std::piecewise_linear_distribution<_RealType1> &__x`)

### 5.981.1 Detailed Description

```
template<typename _RealType = double>
class std::piecewise_linear_distribution<_RealType>
```

A piecewise\_linear\_distribution random number distribution.

The formula for the piecewise linear probability mass function is

Definition at line 5678 of file random.h.

### 5.981.2 Member Typedef Documentation

#### 5.981.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_linear_distribution<_RealType>::result_type
```

The type of the range of the distribution.

Definition at line 5681 of file random.h.

### 5.981.3 Member Function Documentation

#### 5.981.3.1 densities()

```
template<typename _RealType = double>
std::vector<double> std::piecewise_linear_distribution<_RealType>::densities ( ) const [inline]
```

Return a vector of the probability densities of the distribution.

Definition at line 5806 of file random.h.

References std::vector<\_Tp, \_Alloc>::empty().



### 5.981.3.2 intervals()

```
template<typename _RealType = double>
std::vector<_RealType> std::piecewise_linear_distribution< _RealType >::intervals ( ) const
[inline]
```

Return the intervals of the distribution.

Definition at line 5789 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

### 5.981.3.3 max()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5841 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`, and `std::vector< _Tp, _Alloc >::empty()`.

### 5.981.3.4 min()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5831 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::front()`.

### 5.981.3.5 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::piecewise_linear_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 5852 of file random.h.

**5.981.3.6 param()** [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_linear_distribution<_RealType>::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5816 of file random.h.

**5.981.3.7 param()** [2/2]

```
template<typename _RealType = double>
void std::piecewise_linear_distribution<_RealType>::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5824 of file random.h.

**5.981.3.8 reset()**

```
template<typename _RealType = double>
void std::piecewise_linear_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 5782 of file random.h.

**5.981.4 Friends And Related Function Documentation****5.981.4.1 operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::piecewise_linear_distribution<_RealType1> & __x ) [friend]
```

Inserts a piecewise\_linear\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A <code>piecewise_linear_distribution</code> random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**5.981.4.2 `operator==`**

```
template<typename _RealType = double>
bool operator== (
    const piecewise\_linear\_distribution< _RealType > & __d1,
    const piecewise\_linear\_distribution< _RealType > & __d2 ) [friend]
```

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5887 of file `random.h`.

**5.981.4.3 `operator>>`**

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic\_istream<_CharT, _Traits>& operator>> (
    std::basic\_istream< _CharT, _Traits > & __is,
    std::piecewise\_linear\_distribution< _RealType1 > & __x ) [friend]
```

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>piecewise_linear_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.982 std::piecewise\_linear\_distribution&lt; \_RealType &gt;::param\_type Struct Reference

## Public Types

- typedef [piecewise\\_linear\\_distribution](#)< \_RealType > **distribution\_type**

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**param\_type** ([initializer\\_list](#)< \_RealType > \_\_bl, \_Func \_\_fw)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< \_RealType > **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class [piecewise\\_linear\\_distribution](#)< \_RealType >

## 5.982.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_linear_distribution< _RealType >::param_type
```

Parameter type.

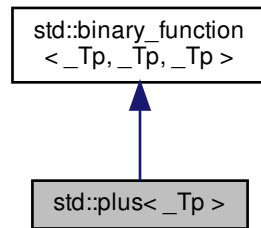
Definition at line 5688 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.983 `std::plus<_Tp>` Struct Template Reference

Inheritance diagram for `std::plus<_Tp>`:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_GLIBCXX14_CONSTEXPR _Tp operator()(const _Tp &__x, const _Tp &__y) const`

#### 5.983.1 Detailed Description

```
template<typename _Tp>  
struct std::plus<_Tp>
```

One of the [math functors](#).

Definition at line 147 of file `stl_function.h`.

#### 5.983.2 Member Typedef Documentation

### 5.983.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

### 5.983.2.2 result\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

### 5.983.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

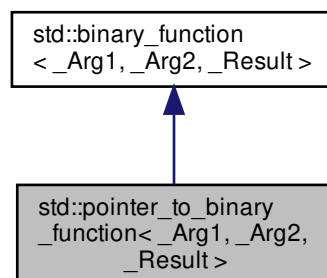
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.984 std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result > Class Template Reference

Inheritance diagram for std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >:



### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- **pointer\_to\_binary\_function** (`_Result(*__x)(_Arg1, _Arg2)`)
- `_Result operator()` (`_Arg1 __x, _Arg2 __y`) const

### Protected Attributes

- `_Result(* M_ptr )(_Arg1, _Arg2)`

#### 5.984.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 1081 of file `stl_function.h`.

#### 5.984.2 Member Typedef Documentation

##### 5.984.2.1 first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary\_function< \_Arg1, \_Arg2, \_Result >::first\_argument\_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 5.984.2.2 result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary\_function< \_Arg1, \_Arg2, \_Result >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

## 5.984.2.3 second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

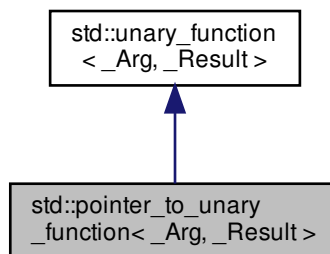
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.985 std::pointer\_to\_unary\_function&lt; \_Arg, \_Result &gt; Class Template Reference

Inheritance diagram for std::pointer\_to\_unary\_function< \_Arg, \_Result >:



## Public Types

- typedef \_Arg [argument\\_type](#)
- typedef \_Result [result\\_type](#)

## Public Member Functions

- **pointer\_to\_unary\_function** (\_Result(\*\_\_x)(\_Arg))
- \_Result **operator()** (\_Arg \_\_x) const

## Protected Attributes

- \_Result(\* **M\_ptr**)(\_Arg)



### 5.985.1 Detailed Description

```
template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 1056 of file `stl_function.h`.

### 5.985.2 Member Typedef Documentation

#### 5.985.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary\_function< _Arg, _Result >::argument\_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.985.2.2 `result_type`

```
template<typename _Arg, typename _Result>
typedef _Result std::unary\_function< _Arg, _Result >::result\_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.986 `std::pointer_traits< _Ptr >` Struct Template Reference

### Public Types

- using [difference\\_type](#) = `__detected_or_t< ptrdiff_t, __difference_type, _Ptr >`
- using [element\\_type](#) = `__detected_or_t< __get_first_arg_t< _Ptr >, __element_type, _Ptr >`
- using [pointer](#) = `_Ptr`
- template<typename \_Up >  
using [rebind](#) = `typename __rebind< _Ptr, _Up >::type`

### Static Public Member Functions

- static \_Ptr **pointer\_to** (\_\_make\_not\_void< [element\\_type](#) > &\_\_e)

#### 5.986.1 Detailed Description

```
template<typename _Ptr>
struct std::pointer_traits<_Ptr>
```

Uniform interface to all pointer-like types.

Definition at line 82 of file ptr\_traits.h.

#### 5.986.2 Member Typedef Documentation

##### 5.986.2.1 difference\_type

```
template<typename _Ptr>
using std::pointer\_traits<_Ptr>::difference\_type = __detected_or_t<ptrdiff_t, __difference_↵
type, _Ptr>
```

The type used to represent the difference between two pointers.

Definition at line 108 of file ptr\_traits.h.

##### 5.986.2.2 element\_type

```
template<typename _Ptr>
using std::pointer\_traits<_Ptr>::element\_type = __detected_or_t<__get_first_arg_t<_Ptr>, __↵
element_type, _Ptr>
```

The type pointed to.

Definition at line 104 of file ptr\_traits.h.

##### 5.986.2.3 pointer

```
template<typename _Ptr>
using std::pointer\_traits<_Ptr>::pointer = _Ptr
```

The pointer type.

Definition at line 100 of file ptr\_traits.h.

#### 5.986.2.4 rebind

```
template<typename _Ptr>
template<typename _Up >
using std::pointer\_traits< _Ptr >::rebind = typename __rebind<_Ptr, _Up>::type
```

A pointer to a different type.

Definition at line 112 of file `ptr_traits.h`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

### 5.987 `std::pointer_traits<_Tp*>` Struct Template Reference

#### Public Types

- typedef ptrdiff\_t [difference\\_type](#)
- typedef \_Tp [element\\_type](#)
- typedef \_Tp \* [pointer](#)
- template<typename \_Up >  
using **rebind** = \_Up \*

#### Static Public Member Functions

- static [pointer pointer\\_to](#) (\_\_make\_not\_void< [element\\_type](#) > &\_\_r) noexcept

#### 5.987.1 Detailed Description

```
template<typename _Tp>
struct std::pointer_traits< _Tp * >
```

Partial specialization for built-in pointers.

Definition at line 127 of file `ptr_traits.h`.

#### 5.987.2 Member Typedef Documentation

#### 5.987.2.1 difference\_type

```
template<typename _Tp >
typedef ptrdiff_t std::pointer_traits< _Tp * >::difference_type
```

Type used to represent the difference between two pointers.

Definition at line 134 of file ptr\_traits.h.

#### 5.987.2.2 element\_type

```
template<typename _Tp >
typedef _Tp std::pointer_traits< _Tp * >::element_type
```

The type pointed to.

Definition at line 132 of file ptr\_traits.h.

#### 5.987.2.3 pointer

```
template<typename _Tp >
typedef _Tp* std::pointer_traits< _Tp * >::pointer
```

The pointer type.

Definition at line 130 of file ptr\_traits.h.

### 5.987.3 Member Function Documentation

#### 5.987.3.1 pointer\_to()

```
template<typename _Tp >
static pointer std::pointer_traits< _Tp * >::pointer_to (
    __make_not_void< element_type > & __r ) [inline], [static], [noexcept]
```

Obtain a pointer to an object.

## Parameters

↩	A reference to an object of type <code>element_type</code>
↩	
↩	
↩	
<i>r</i>	

## Returns

`addressof(__r)`

Definition at line 145 of file `ptr_traits.h`.

References `std::addressof()`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

## 5.988 `std::poisson_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- **`poisson_distribution`** (`double __mean=1.0`)
- **`poisson_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate (result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- [result\\_type](#) `max` () const
- `double` `mean` () const
- [result\\_type](#) `min` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- [param\\_type](#) `param` () const
- `void` `param` (const [param\\_type](#) &\_\_param)
- `void` `reset` ()

## Friends

- template<typename \_IntType1 , typename \_CharT , typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::poisson\_distribution< \_IntType1 > &\_\_x)
- bool operator== (const poisson\_distribution &\_\_d1, const poisson\_distribution &\_\_d2)
- template<typename \_IntType1 , typename \_CharT , typename \_Traits >  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::poisson\_distribution< \_IntType1 > &\_\_x)

## 5.988.1 Detailed Description

```
template<typename _IntType = int>
class std::poisson_distribution< _IntType >
```

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Definition at line 4330 of file random.h.

## 5.988.2 Member Typedef Documentation

## 5.988.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::poisson_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 4333 of file random.h.

## 5.988.3 Member Function Documentation

## 5.988.3.1 max()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4429 of file random.h.

References std::numeric\_limits< \_Tp >::max().

**5.988.3.2 mean()**

```
template<typename _IntType = int>
double std::poisson_distribution< _IntType >::mean ( ) const [inline]
```

Returns the distribution parameter mean.

Definition at line 4400 of file random.h.

**5.988.3.3 min()**

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4422 of file random.h.

**5.988.3.4 operator>() [1/2]**

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::poisson_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 4437 of file random.h.

**5.988.3.5 operator>() [2/2]**

```
template<typename _IntType >
template<typename _UniformRandomNumberGenerator >
poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator()
(
    _UniformRandomNumberGenerator & __urng,
    const param_type & __param )
```

A rejection algorithm when mean  $\geq 12$  and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1281 of file bits/random.tcc.

References `std::abs()`, `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::numeric_limits< _Tp >::max()`.

**5.988.3.6 param()** [1/2]

```
template<typename _IntType = int>
param_type std::poisson_distribution< _IntType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4407 of file random.h.

**5.988.3.7 param()** [2/2]

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4415 of file random.h.

**5.988.3.8 reset()**

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 4393 of file random.h.

References `std::normal_distribution<_RealType>::reset()`.

**5.988.4 Friends And Related Function Documentation****5.988.4.1 operator<<**

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::poisson_distribution< _IntType1 > & __x ) [friend]
```

Inserts a `poisson_distribution` random number distribution `__x` into the output stream `__os`.



**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A <code>poisson_distribution</code> random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**5.988.4.2 operator==**

```
template<typename _IntType = int>
bool operator== (
    const poisson_distribution< _IntType > & __d1,
    const poisson_distribution< _IntType > & __d2 ) [friend]
```

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4473 of file random.h.

**5.988.4.3 operator>>**

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::poisson_distribution< _IntType1 > & __x ) [friend]
```

Extracts a `poisson_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A <code>poisson_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.989 std::poisson\_distribution< \_IntType >::param\_type Struct Reference

### Public Types

- typedef [poisson\\_distribution](#)< \_IntType > **distribution\_type**

### Public Member Functions

- param\_type** (double \_\_mean=1.0)
- double **mean** () const

### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **poisson\_distribution**< \_IntType >

#### 5.989.1 Detailed Description

```
template<typename _IntType = int>
struct std::poisson_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4340 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.990 std::priority\_queue< \_Tp, \_Sequence, \_Compare > Class Template Reference

### Public Types

- typedef \_Sequence::const\_reference **const\_reference**
- typedef \_Sequence **container\_type**
- typedef \_Sequence::reference **reference**
- typedef \_Sequence::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Sequence::value\_type **value\_type**

## Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and<is_default_constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>`  
`priority_queue ()`
- `priority_queue (const _Compare &__x, const _Sequence &__s)`
- `priority_queue (const _Compare &__x, _Sequence && __s=_Sequence())`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`priority_queue (const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`priority_queue (const _Compare &__x, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`priority_queue (const _Compare &__x, const _Sequence &__c, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`priority_queue (const _Compare &__x, _Sequence && __c, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`priority_queue (const priority_queue &__q, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`priority_queue (priority_queue && __q, const _Alloc &__a)`
- `template<typename _InputIterator >`  
`priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s)`
- `template<typename _InputIterator >`  
`priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare(), _Sequence && __s=_Sequence())`
- `template<typename... _Args>`  
`void emplace (_Args &&... __args)`
- `bool empty () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type && __x)`
- `size_type size () const`
- `void swap (priority_queue &__pq) noexcept(__and<__c++1z or gnu++11 __is_nothrow_swappable<_Sequence>, __is_nothrow_swappable<_Compare>>::value)`
- `const_reference top () const`

## Protected Attributes

- `_Sequence c`
- `_Compare comp`

## 5.990.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue<_Tp, _Sequence, _Compare>
```

A standard container automatically sorting its contents.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>vector&lt;_Tp&gt;</code> .
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Sequence::value_type&gt;</code> .

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

## Note

No equality/comparison operators are provided for `priority_queue`.

Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 435 of file `stl_queue.h`.

## 5.990.2 Constructor &amp; Destructor Documentation

5.990.2.1 `priority_queue()` [1/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and<is_default_constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>
std::priority_queue<_Tp, _Sequence, _Compare >::priority_queue ( ) [inline]
```

Default constructor creates no elements.

Definition at line 485 of file `stl_queue.h`.

### 5.990.2.2 `priority_queue()` [2/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
template<typename _InputIterator >
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __x,
    const _Sequence & __s ) [inline]
```

Builds a queue from a range.

#### Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	A comparison functor describing a strict weak ordering.
<code>__s</code>	An initial sequence with which to start.

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 557 of file `stl_queue.h`.

## 5.990.3 Member Function Documentation

### 5.990.3.1 `empty()`

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
bool std::priority_queue< _Tp, _Sequence, _Compare >::empty ( ) const [inline]
```

Returns true if the queue is empty.

Definition at line 583 of file `stl_queue.h`.

### 5.990.3.2 pop()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵  
Sequence::value_type>>  
void std::priority_queue<_Tp, _Sequence, _Compare >::pop ( ) [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 646 of file stl\_queue.h.

### 5.990.3.3 push()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵  
Sequence::value_type>>  
void std::priority_queue<_Tp, _Sequence, _Compare >::push (   
    const value_type & __x ) [inline]
```

Add data to the queue.

#### Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 611 of file stl\_queue.h.

References std::push\_heap().

### 5.990.3.4 size()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵  
Sequence::value_type>>  
size_type std::priority_queue<_Tp, _Sequence, _Compare >::size ( ) const [inline]
```

Returns the number of elements in the queue.

Definition at line 588 of file stl\_queue.h.

### 5.990.3.5 top()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
const_reference std::priority_queue< _Tp, _Sequence, _Compare >::top ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 596 of file `stl_queue.h`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.991 std::promise< \_Res > Class Template Reference

### Public Member Functions

- **promise** ([promise](#) &&\_\_rhs) noexcept
- template<typename \_Allocator >  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &\_\_a)
- template<typename \_Allocator >  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &, [promise](#) &&\_\_rhs)
- **promise** (const [promise](#) &)=delete
- [future](#)< \_Res > **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** (const [promise](#) &)=delete
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** (const \_Res &\_\_r)
- void **set\_value** (\_Res &&\_\_r)
- void **set\_value\_at\_thread\_exit** (const \_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &&\_\_r)
- void **swap** ([promise](#) &&\_\_rhs) noexcept

### Friends

- template<typename , typename >  
class **\_State::Setter**

### 5.991.1 Detailed Description

```
template<typename _Res>
class std::promise< _Res >
```

Primary template for promise.

Definition at line 134 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

5.992 `std::promise<_Res &>` Class Template Reference

## Public Member Functions

- **promise** ([promise](#) &&\_\_rhs) noexcept
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &\_\_a)
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &, [promise](#) &&\_\_rhs)
- **promise** (const [promise](#) &)=delete
- [future](#)<\_Res &> **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** (const [promise](#) &)=delete
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** (\_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &\_\_r)
- void **swap** ([promise](#) &\_\_rhs) noexcept

## Friends

- `template<typename , typename >`  
class **\_State::Setter**

## 5.992.1 Detailed Description

```
template<typename _Res>
class std::promise<_Res &>
```

Partial specialization for `promise<R&>`

Definition at line 1154 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

5.993 `std::promise< void >` Class Template Reference

## Public Member Functions

- **promise** ([promise](#) &&\_\_rhs) noexcept
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &\_\_a)
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &, [promise](#) &&\_\_rhs)
- **promise** (const [promise](#) &)=delete
- [future](#)< void > **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** (const [promise](#) &)=delete
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** ()
- void **set\_value\_at\_thread\_exit** ()
- void **swap** ([promise](#) &\_\_rhs) noexcept



## Friends

- `template<typename , typename >`  
`class _State::_Setter`

## 5.993.1 Detailed Description

`template<>`  
`class std::promise< void >`

Explicit specialization for `promise<void>`

Definition at line 1244 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

5.994 `std::queue< _Tp, _Sequence >` Class Template Reference

## Public Types

- `typedef _Sequence::const_reference const_reference`
- `typedef _Sequence container_type`
- `typedef _Sequence::reference reference`
- `typedef _Sequence::size_type size_type`
- `typedef _Sequence::value_type value_type`

## Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>`  
`queue ()`
- `queue (const _Sequence &__c)`
- `queue (_Sequence &&__c)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>`  
`queue (const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>`  
`queue (const _Sequence &__c, const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>`  
`queue (_Sequence &&__c, const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>`  
`queue (const queue &__q, const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>`  
`queue (queue &&__q, const _Alloc &__a)`
- `reference back ()`
- `const_reference back () const`
- `template<typename... _Args>`  
`void emplace (_Args &&... __args)`
- `bool empty () const`
- `reference front ()`
- `const_reference front () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type &&__x)`
- `size_type size () const`
- `void swap (queue &__q) noexcept(__is_nothrow_swappable<_Sequence>::value)`

## Protected Attributes

- `_Sequence` [c](#)

## Friends

- `template<typename _Tp1, typename _Seq1 >`  
`bool operator< (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >`  
`bool operator== (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`

## 5.994.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue< _Tp, _Sequence >
```

A standard container giving FIFO behavior.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> .

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 96 of file `stl_queue.h`.

## 5.994.2 Constructor &amp; Destructor Documentation

5.994.2.1 `queue()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↔
_Seq>::value>::type>
std::queue< _Tp, _Sequence >::queue ( ) [inline]
```

Default constructor creates no elements.

Definition at line 152 of file `stl_queue.h`.

### 5.994.3 Member Function Documentation

#### 5.994.3.1 `back()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::back ( ) [inline]
```

Returns a read/write reference to the data at the last element of the queue.

Definition at line 224 of file `stl_queue.h`.

#### 5.994.3.2 `back()` [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::back ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 235 of file `stl_queue.h`.

#### 5.994.3.3 `empty()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue< _Tp, _Sequence >::empty ( ) const [inline]
```

Returns true if the queue is empty.

Definition at line 189 of file `stl_queue.h`.

References `std::queue`< \_Tp, \_Sequence >::c.

#### 5.994.3.4 `front()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::front ( ) [inline]
```

Returns a read/write reference to the data at the first element of the queue.

Definition at line 202 of file `stl_queue.h`.

### 5.994.3.5 front() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::front ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 213 of file stl\_queue.h.

### 5.994.3.6 pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue<_Tp, _Sequence>::pop ( ) [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.


Definition at line 284 of file stl\_queue.h.

### 5.994.3.7 push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue<_Tp, _Sequence>::push (
    const value_type & __x ) [inline]
```

Add data to the end of the queue.

#### Parameters

 _x	Data to be added.
--	-------------------

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 251 of file stl\_queue.h.

References std::queue<\_Tp, \_Sequence>::c.

#### 5.994.3.8 size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size ( ) const [inline]
```

Returns the number of elements in the queue.

Definition at line 194 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

### 5.994.4 Member Data Documentation

#### 5.994.4.1 c

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequence std::queue< _Tp, _Sequence >::c [protected]
```

`c` is the underlying container.

Definition at line 139 of file `stl_queue.h`.

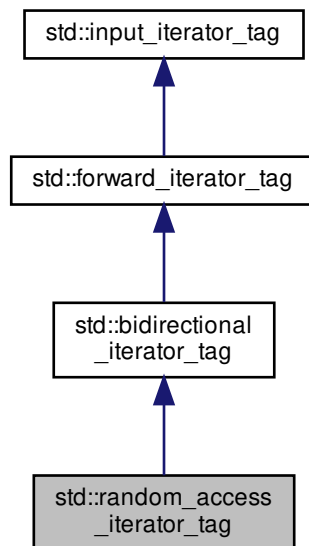
Referenced by `std::queue< _Tp, _Sequence >::empty()`, `std::operator<()`, `std::operator==()`, `std::queue< _Tp, _Sequence >::push()`, and `std::queue< _Tp, _Sequence >::size()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.995 `std::random_access_iterator_tag` Struct Reference

Inheritance diagram for `std::random_access_iterator_tag`:



### 5.995.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations.

Definition at line 103 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.996 `std::random_device` Class Reference

### Public Types

- typedef unsigned int [result\\_type](#)

### Public Member Functions

- **random\_device** (const [std::string](#) &\_\_token="mt19937")
- **random\_device** (const [random\\_device](#) &)=delete
- double **entropy** () const noexcept
- [result\\_type](#) **operator()** ()
- void **operator=** (const [random\\_device](#) &)=delete

### Static Public Member Functions

- static constexpr [result\\_type](#) **max** ()
- static constexpr [result\\_type](#) **min** ()

#### 5.996.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1560 of file random.h.

#### 5.996.2 Member Typedef Documentation

##### 5.996.2.1 [result\\_type](#)

```
typedef unsigned int std::random\_device::result\_type
```

The type of the generated random value.

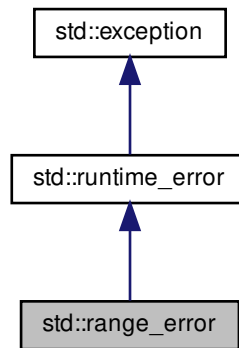
Definition at line 1564 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.997 `std::range_error` Class Reference

Inheritance diagram for `std::range_error`:



### Public Member Functions

- **`range_error`** (const [string](#) &\_\_arg) `_GLIBCXX_TXN_SAFE`
- **`range_error`** (const char \*) `_GLIBCXX_TXN_SAFE`
- virtual const char \* [what](#) () const `_GLIBCXX_TXN_SAFE_DYN noexcept`

#### 5.997.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 230 of file `stdexcept`.

#### 5.997.2 Member Function Documentation

##### 5.997.2.1 `what()`

```
virtual const char* std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

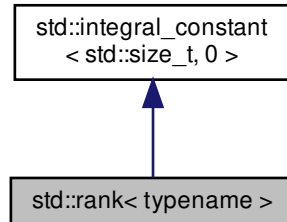
The documentation for this class was generated from the following file:

- [stdexcept](#)



### 5.998 `std::rank< typename >` Struct Template Reference

Inheritance diagram for `std::rank< typename >`:



#### Public Types

- typedef [integral\\_constant](#)< `std::size_t`, `__v` > **type**
- typedef `std::size_t` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `std::size_t` **value**

#### 5.998.1 Detailed Description

```
template<typename>  
struct std::rank< typename >
```

rank

Definition at line 1244 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.999 std::ratio&lt; \_Num, \_Den &gt; Struct Template Reference

## Public Types

- typedef [ratio](#)< num, den > **type**

## Static Public Attributes

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

## 5.999.1 Detailed Description

```
template<intmax_t _Num, intmax_t _Den = 1>
struct std::ratio< _Num, _Den >
```

Provides compile-time rational arithmetic.

This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type intmax\_t. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

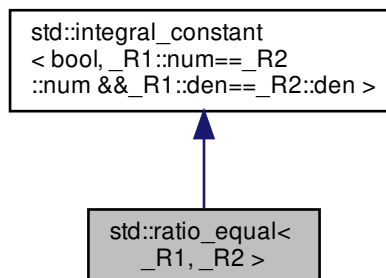
Definition at line 263 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.1000 std::ratio\_equal&lt; \_R1, \_R2 &gt; Struct Template Reference

Inheritance diagram for std::ratio\_equal< \_R1, \_R2 >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr bool **value**

### 5.1000.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_equal< _R1, _R2 >
```

ratio\_equal

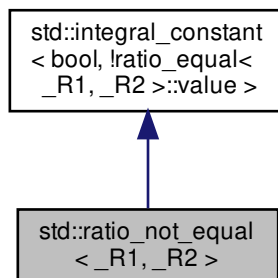
Definition at line 340 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

### 5.1001 std::ratio\_not\_equal< \_R1, \_R2 > Struct Template Reference

Inheritance diagram for std::ratio\_not\_equal< \_R1, \_R2 >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 5.1001.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_not_equal< _R1, _R2 >
```

ratio\_not\_equal

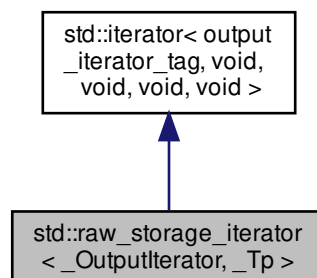
Definition at line 346 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.1002 std::raw\_storage\_iterator&lt; \_OutputIterator, \_Tp &gt; Class Template Reference

Inheritance diagram for std::raw\_storage\_iterator< \_OutputIterator, \_Tp >:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- **raw\_storage\_iterator** ([\\_OutputIterator](#) \_\_x)
- [\\_OutputIterator](#) **base** () const
- [raw\\_storage\\_iterator](#) & **operator\*** ()
- [raw\\_storage\\_iterator](#) & **operator++** ()
- [raw\\_storage\\_iterator](#) **operator++** (int)
- [raw\\_storage\\_iterator](#) & **operator=** (const [\\_Tp](#) &\_\_element)
- [raw\\_storage\\_iterator](#) & **operator=** ([\\_Tp](#) &&\_\_element)

### Protected Attributes

- [\\_OutputIterator](#) **\_M\_iter**

#### 5.1002.1 Detailed Description

```
template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator< _OutputIterator, _Tp >
```

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 68 of file `stl_raw_storage_iter.h`.

#### 5.1002.2 Member Typedef Documentation

##### 5.1002.2.1 [difference\\_type](#)

```
typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference\_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

#### 5.1002.2.2 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >←  
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file stl\_iterator\_base\_types.h.

#### 5.1002.2.3 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 127 of file stl\_iterator\_base\_types.h.

#### 5.1002.2.4 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 129 of file stl\_iterator\_base\_types.h.

#### 5.1002.2.5 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file stl\_iterator\_base\_types.h.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

### 5.1003 std::recursive\_mutex Class Reference

Inherits std::\_\_recursive\_mutex\_base.

### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

### Public Member Functions

- **recursive\_mutex** (const [recursive\\_mutex](#) &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- [recursive\\_mutex](#) & **operator=** (const [recursive\\_mutex](#) &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

### Private Types

- typedef \_\_gthread\_recursive\_mutex\_t **\_\_native\_type**

### Private Attributes

- \_\_native\_type **\_M\_mutex**

## 5.1003.1 Detailed Description

The standard recursive mutex type.

Definition at line 93 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.1004 std::recursive\_timed\_mutex Class Reference

### Public Member Functions

- **recursive\_timed\_mutex** (const [recursive\\_timed\\_mutex](#) &)=delete
- void **lock** ()
- [recursive\\_timed\\_mutex](#) & **operator=** (const [recursive\\_timed\\_mutex](#) &)=delete
- bool **try\_lock** ()
- template<typename \_Rep , typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock , typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

## 5.1004.1 Detailed Description

`recursive_timed_mutex`

Definition at line 363 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

5.1005 `std::reference_wrapper<_Tp>` Class Template Reference

Inherits `std::_Reference_wrapper_base_memfun<_Tp, bool>`.

## Public Types

- `typedef _Tp type`

## Public Member Functions

- `reference_wrapper` (`_Tp &__indata`) `noexcept`
- `reference_wrapper` (`_Tp &&__indata`) `=delete`
- `reference_wrapper` (`const reference_wrapper &__indata`) `=default`
- `_Tp &get` () `const noexcept`
- `operator _Tp &` () `const noexcept`
- `template<typename... _Args>`  
`result_of<_Tp &(_Args &&...)>::type operator()` (`_Args &&... __args`) `const`
- `reference_wrapper &operator=` (`const reference_wrapper &__indata`) `=default`

## 5.1005.1 Detailed Description

```
template<typename _Tp>
class std::reference_wrapper<_Tp>
```

Primary class template for `reference_wrapper`.

Definition at line 1880 of file `type_traits`.

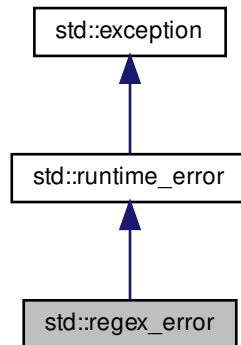
The documentation for this class was generated from the following files:

- [type\\_traits](#)
- [refwrap.h](#)



## 5.1006 std::regex\_error Class Reference

Inheritance diagram for std::regex\_error:



### Public Member Functions

- [regex\\_error](#) ([regex\\_constants::error\\_type](#) \_\_ecode)
- [regex\\_constants::error\\_type](#) code () const
- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

### Friends

- void [\\_\\_throw\\_regex\\_error](#) ([regex\\_constants::error\\_type](#), const char \*)

### 5.1006.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 132 of file `regex_error.h`.

### 5.1006.2 Constructor & Destructor Documentation

#### 5.1006.2.1 regex\_error()

```
std::regex_error::regex_error (  
    regex\_constants::error\_type __ecode ) [explicit]
```

Constructs a `regex_error` object.

## Parameters

<code>__ecode</code>	the regex error code.
----------------------	-----------------------

## 5.1006.3 Member Function Documentation

5.1006.3.1 `code()`

```
regex_constants::error_type std::regex_error::code ( ) const [inline]
```

Gets the regex error code.

## Returns

the regex error code.

Definition at line 153 of file `regex_error.h`.

5.1006.3.2 `what()`

```
virtual const char* std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

5.1007 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

## Public Types

- `typedef std::ptrdiff_t` **difference\_type**
- `typedef` [std::forward\\_iterator\\_tag](#) **iterator\_category**
- `typedef const` [value\\_type](#) \* **pointer**
- `typedef const` [value\\_type](#) & **reference**
- `typedef` [basic\\_regex< \\_Ch\\_type, \\_Rx\\_traits >](#) **regex\_type**
- `typedef` [match\\_results< \\_Bi\\_iter >](#) **value\_type**

## Public Member Functions

- `regex_iterator` ()
- `regex_iterator` (`_Bi_iter __a`, `_Bi_iter __b`, `const regex_type &__re`, `regex_constants::match_flag_type __m`↵  
`m=regex_constants::match_default`)
- `regex_iterator` (`_Bi_iter`, `_Bi_iter`, `const regex_type &&`, `regex_constants::match_flag_type=regex_constants::match_default`)=delete
- `regex_iterator` (`const regex_iterator &__rhs`)=default
- `bool operator!=` (`const regex_iterator &__rhs`) const
- `const value_type & operator*` () const
- `regex_iterator & operator++` ()
- `regex_iterator operator++` (int)
- `const value_type * operator->` () const
- `regex_iterator & operator=` (`const regex_iterator &__rhs`)=default
- `bool operator==` (`const regex_iterator &__rhs`) const

## 5.1007.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2479 of file `regex.h`.

## 5.1007.2 Constructor &amp; Destructor Documentation

5.1007.2.1 `regex_iterator()` [1/3]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( ) [inline]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Definition at line 2493 of file `regex.h`.





Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator()`.

5.1007.2.2 `regex_iterator()` [2/3]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a `regex_iterator`...

## Parameters

<a href="#"></a> <code>_a</code>	[IN] The start of a text range to search.
<a href="#"></a> <code>_b</code>	[IN] One-past-the-end of the text range to search.
<a href="#"></a> <code>_re</code>	[IN] The regular expression to match.
<a href="#"></a> <code>_m</code>	[IN] Policy flags for match rules.

Definition at line 2504 of file regex.h.

References `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator()`, and `std::regex_search()`.

## 5.1007.2.3 regex\_iterator() [3/3]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
    const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) [default]
```

Copy constructs a `regex_iterator`.

## 5.1007.3 Member Function Documentation

## 5.1007.3.1 operator!=(())

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=( (
    const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) const [inline]
```

Tests the inequivalence of two `regex_iterator`s.

Definition at line 2539 of file regex.h.

## 5.1007.3.2 operator\*()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ( ) const
[inline]
```

Dereferences a `regex_iterator`.

Definition at line 2546 of file regex.h.

**5.1007.3.3 operator++()** [1/2]

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ( )
```

Increments a regex\_iterator.

Definition at line 519 of file regex.tcc.

References std::regex\_constants::match\_continuous, std::regex\_constants::match\_not\_null, and std::regex\_search().

**5.1007.3.4 operator++()** [2/2]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
    int ) [inline]
```

Postincrements a regex\_iterator.

Definition at line 2566 of file regex.h.

**5.1007.3.5 operator->()**

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type* std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ( ) const
[inline]
```

Selects a regex\_iterator member.

Definition at line 2553 of file regex.h.

**5.1007.3.6 operator=()**

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
    const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) [default]
```

Assigns one regex\_iterator to another.

## 5.1007.3.7 operator==( )

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator==(
    const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) const
```

Tests the equivalence of two regex iterators.

Definition at line 503 of file regex.tcc.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

## 5.1008 std::regex\_token\_iterator&lt; \_Bi\_iter, \_Ch\_type, \_Rx\_traits &gt; Class Template Reference

## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef const [value\\_type](#) \* **pointer**
- typedef const [value\\_type](#) & **reference**
- typedef [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > **regex\_type**
- typedef [sub\\_match](#)< \_Bi\_iter > **value\_type**

## Public Member Functions

- [regex\\_token\\_iterator](#) ( )
- [regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, int \_\_submatch=0, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, const [std::vector](#)< int > &\_\_submatches, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, [initializer\\_list](#)< int > \_\_submatches, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- template<std::size\_t \_Nm>  
[regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, const int(&\_\_submatches)[\_Nm], [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** ( \_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, int=0, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** ( \_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, const [std::vector](#)< int > &, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** ( \_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, [initializer\\_list](#)< int >, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- template<std::size\_t \_Nm>  
**regex\_token\_iterator** ( \_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, const int(&)[\_Nm], [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) (const [regex\\_token\\_iterator](#) &\_\_rhs)
- bool **operator!=** (const [regex\\_token\\_iterator](#) &\_\_rhs) const
- const [value\\_type](#) & **operator\*** ( ) const
- [regex\\_token\\_iterator](#) & **operator++** ( )
- [regex\\_token\\_iterator](#) **operator++** (int)
- const [value\\_type](#) \* **operator->** ( ) const
- [regex\\_token\\_iterator](#) & **operator=** (const [regex\\_token\\_iterator](#) &\_\_rhs)
- bool **operator==** (const [regex\\_token\\_iterator](#) &\_\_rhs) const

### 5.1008.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a `std::sub_match` object.

Definition at line 2599 of file `regex.h`.

### 5.1008.2 Constructor & Destructor Documentation

#### 5.1008.2.1 `regex_token_iterator()` [1/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( ) [inline]
```

Default constructs a `regex_token_iterator`.

A default-constructed `regex_token_iterator` is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

Definition at line 2617 of file `regex.h`.

#### 5.1008.2.2 `regex_token_iterator()` [2/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    int __submatch = 0,
    regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a `regex_token_iterator`...

#### Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatch</code>	[IN] Which submatch to return. There are some special values for this parameter: <span style="float: right;">Generated by Doxygen</span> <ul style="list-style-type: none"> <li>-1 each enumerated subexpression does NOT match the regular expression (aka field splitting)</li> </ul>

Definition at line 2639 of file regex.h.

### 5.1008.2.3 regex\_token\_iterator() [3/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    const std::vector< int > & __submatches,
    regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a regex\_token\_iterator...

#### Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2655 of file regex.h.

### 5.1008.2.4 regex\_token\_iterator() [4/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    initializer_list< int > __submatches,
    regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a regex\_token\_iterator...

#### Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.



Definition at line 2672 of file regex.h.

#### 5.1008.2.5 regex\_token\_iterator() [5/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
template<std::size_t _Nm>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    _Bi_iter __a,
    _Bi_iter __b,
    const regex_type & __re,
    const int (&) __submatches[_Nm],
    regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a regex\_token\_iterator...

##### Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2690 of file regex.h.

#### 5.1008.2.6 regex\_token\_iterator() [6/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
    const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) [inline]
```

Copy constructs a regex\_token\_iterator.

##### Parameters

<code>__rhs</code>	[IN] A regex_token_iterator to copy.
--------------------	--------------------------------------

Definition at line 2722 of file regex.h.

#### 5.1008.3 Member Function Documentation

### 5.1008.3.1 operator!=(())

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=(
    const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) const [inline]
```

Compares a regex\_token\_iterator to another for inequality.

Definition at line 2744 of file regex.h.

### 5.1008.3.2 operator\*()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type& std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ( )
const [inline]
```

Dereferences a regex\_token\_iterator.

Definition at line 2751 of file regex.h.

### 5.1008.3.3 operator++() [1/2]

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ( )
```

Increments a regex\_token\_iterator.

Definition at line 614 of file regex.tcc.

### 5.1008.3.4 operator++() [2/2]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
    int ) [inline]
```

Postincrements a regex\_token\_iterator.

Definition at line 2771 of file regex.h.

#### 5.1008.3.5 operator->()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type* std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ( )
const [inline]
```

Selects a `regex_token_iterator` member.

Definition at line 2758 of file `regex.h`.

#### 5.1008.3.6 operator=()

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
    const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )
```

Assigns a `regex_token_iterator` to another.

##### Parameters

<code>__rhs</code>	[IN] A <code>regex_token_iterator</code> to copy.
--------------------	---

Definition at line 578 of file `regex.tcc`.

#### 5.1008.3.7 operator==( )

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
    const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) const
```

Compares a `regex_token_iterator` to another for equality.

Definition at line 594 of file `regex.tcc`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

### 5.1009 std::regex\_traits< \_Ch\_type > Class Template Reference

#### Public Types

- typedef `_RegexMask` **char\_class\_type**
- typedef `_Ch_type` **char\_type**
- typedef `std::locale` **locale\_type**
- typedef `std::basic_string< char_type >` **string\_type**

## Public Member Functions

- [regex\\_traits](#) ()
- [locale\\_type](#) [getloc](#) () const
- [locale\\_type](#) [imbue](#) ([locale\\_type](#) \_\_loc)
- bool [isctype](#) (\_Ch\_type \_\_c, char\_class\_type \_\_f) const
- template<typename \_Fwd\_iter >  
char\_class\_type [lookup\\_classname](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last, bool \_\_icase=false) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [lookup\\_collatename](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [transform](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [transform\\_primary](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- char\_type [translate](#) (char\_type \_\_c) const
- char\_type [translate\\_nocase](#) (char\_type \_\_c) const
- int [value](#) (\_Ch\_type \_\_ch, int \_\_radix) const

## Static Public Member Functions

- static std::size\_t [length](#) (const char\_type \*\_\_p)

## Protected Attributes

- [locale\\_type](#) [\\_M\\_locale](#)

## 5.1009.1 Detailed Description

```
template<typename _Ch_type>
class std::regex_traits< _Ch_type >
```

Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class `regex` is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 81 of file `regex.h`.

## 5.1009.2 Constructor &amp; Destructor Documentation

### 5.1009.2.1 `regex_traits()`

```
template<typename _Ch_type>
std::regex_traits< _Ch_type >::regex_traits ( ) [inline]
```

Constructs a default traits object.

Definition at line 158 of file `regex.h`.

## 5.1009.3 Member Function Documentation

### 5.1009.3.1 `getloc()`

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::getloc ( ) const [inline]
```

Gets a copy of the current locale in use by the `regex_traits` object.

Definition at line 371 of file `regex.h`.

### 5.1009.3.2 `imbue()`

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::imbue (
    locale_type __loc ) [inline]
```

Imbues the `regex_traits` object with a copy of a new locale.

#### Parameters

<code>__loc</code>	A locale.
--------------------	-----------

#### Returns

a copy of the previous locale in use by the `regex_traits` object.

#### Note

Calling `imbue` with a different locale than the one currently in use invalidates all cached data held by `*this`.

Definition at line 360 of file `regex.h`.

### 5.1009.3.3 isctype()

```
template<typename _Ch_type>
bool std::regex_traits< _Ch_type >::isctype (
    _Ch_type __c,
    char_class_type __f ) const
```

Determines if `c` is a member of an identified class.

#### Parameters

<code>__c</code>	a character.
<code>__f</code>	a class type (as returned from <code>lookup_classname</code> ).

#### Returns

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

#### Exceptions

<code>std::bad_cast</code>	if the current locale does not have a ctype facet.
----------------------------	--

Definition at line 327 of file `regex.tcc`.

### 5.1009.3.4 length()

```
template<typename _Ch_type>
static std::size_t std::regex_traits< _Ch_type >::length (
    const char_type * __p ) [inline], [static]
```

Gives the length of a C-style string starting at `__p`.

#### Parameters

<code>__p</code>	a pointer to the start of a character sequence.
------------------	---

#### Returns

the number of characters between `*__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 171 of file `regex.h`.

### 5.1009.3.5 lookup\_classname()

```
template<typename _Ch_type >
template<typename _Fwd_iter >
regex_traits< _Ch_type >::char_class_type std::regex_traits< _Ch_type >::lookup_classname (
    _Fwd_iter __first,
    _Fwd_iter __last,
    bool __icase = false ) const
```

Maps one or more characters to a named character classification.

#### Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.
<code>__icase</code>	ignores the case of the classification name.

#### Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- d
- w
- s
- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower
- print
- punct
- space
- upper
- xdigit

Definition at line 283 of file `regex.tcc`.

## 5.1009.3.6 lookup\_collatename()

```
template<typename _Ch_type >
template<typename _Fwd_iter >
regex_traits< _Ch_type >::string_type std::regex_traits< _Ch_type >::lookup_collatename (
    _Fwd_iter __first,
    _Fwd_iter __last ) const
```

Gets a collation element by name.

## Parameters

<i>__first</i>	beginning of the collation element name.
<i>__last</i>	one-past-the-end of the collation element name.

## Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [*\_\_first*, *\_\_last*). Returns an empty string if the character sequence is not a valid collating element.

Definition at line 127 of file regex.tcc.

## 5.1009.3.7 transform()

```
template<typename _Ch_type>
template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::transform (
    _Fwd_iter __first,
    _Fwd_iter __last ) const [inline]
```

Gets a sort key for a character sequence.

## Parameters

<i>__first</i>	beginning of the character sequence.
<i>__last</i>	one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [*F1*, *F2*) such that if the character sequence [*G1*, *G2*) sorts before the character sequence [*H1*, *H2*) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

## Returns

a locale-specific sort key equivalent to the input range.



**Exceptions**

<code>std::bad_cast</code>	if the current locale does not have a collate facet.
----------------------------	--

Definition at line 224 of file regex.h.

Referenced by `std::regex_traits<_CharType>::transform_primary()`.

**5.1009.3.8 transform\_primary()**

```
template<typename _Ch_type>
template<typename _Fwd_iter >
string_type std::regex_traits<_Ch_type>::transform_primary (
    _Fwd_iter __first,
    _Fwd_iter __last ) const [inline]
```

Gets a sort key for a character sequence, independent of case.

**Parameters**

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type>>) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

**Todo** Implement this function correctly.

Definition at line 248 of file regex.h.

**5.1009.3.9 translate()**

```
template<typename _Ch_type>
char_type std::regex_traits<_Ch_type>::translate (
    char_type __c ) const [inline]
```

Performs the identity translation.

**Parameters**

<code>__c</code>	A character to the locale-specific character set.
------------------	---

**Returns**

\_\_c.

Definition at line 182 of file regex.h.

**5.1009.3.10 translate\_nocase()**

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate_nocase (
    char_type __c ) const [inline]
```

Translates a character into a case-insensitive equivalent.

**Parameters**

<code>__c</code>	A character to the locale-specific character set.
------------------	---

**Returns**

the locale-specific lower-case equivalent of \_\_c.

**Exceptions**

<code>std::bad_cast</code>	if the imbued locale does not support the ctype facet.
----------------------------	--

Definition at line 195 of file regex.h.

**5.1009.3.11 value()**

```
template<typename _Ch_type>
int std::regex_traits< _Ch_type >::value (
    _Ch_type __ch,
    int __radix ) const
```

Converts a digit to an int.

**Parameters**

<code>__ch</code>	a character representing a digit.
<code>__radix</code>	the radix if the numeric conversion (limited to 8, 10, or 16).

**Returns**

the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

Definition at line 341 of file `regex.tcc`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

**5.1010 `std::remove_all_extents<_Tp>` Struct Template Reference****Public Types**

- `typedef _Tp type`

**5.1010.1 Detailed Description**

```
template<typename _Tp>
struct std::remove_all_extents<_Tp>
```

`remove_all_extents`

Definition at line 762 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.1011 `std::remove_const<_Tp>` Struct Template Reference****Public Types**

- `typedef _Tp type`

**5.1011.1 Detailed Description**

```
template<typename _Tp>
struct std::remove_const<_Tp>
```

`remove_const`

Definition at line 1329 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.1012 `std::remove_cv<_Tp>` Struct Template Reference

### Public Types

- typedef `remove_const<typename remove_volatile<_Tp>::type>::type` **type**

#### 5.1012.1 Detailed Description

```
template<typename _Tp>  
struct std::remove_cv<_Tp>
```

`remove_cv`

Definition at line 190 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.1013 `std::remove_extent<_Tp>` Struct Template Reference

### Public Types

- typedef `_Tp` **type**

#### 5.1013.1 Detailed Description

```
template<typename _Tp>  
struct std::remove_extent<_Tp>
```

`remove_extent`

Definition at line 1696 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.1014 `std::remove_pointer<_Tp>` Struct Template Reference

Inherits `std::__remove_pointer_helper<_Tp, typename>`.

#### Public Types

- `typedef _Tp type`

#### 5.1014.1 Detailed Description

```
template<typename _Tp>  
struct std::remove_pointer<_Tp >
```

`remove_pointer`

Definition at line 1742 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.1015 `std::remove_reference<_Tp >` Struct Template Reference

#### Public Types

- `typedef _Tp type`

#### 5.1015.1 Detailed Description

```
template<typename _Tp>  
struct std::remove_reference<_Tp >
```

`remove_reference`

Definition at line 1404 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 5.1016 `std::remove_volatile<_Tp >` Struct Template Reference

#### Public Types

- `typedef _Tp type`

## 5.1016.1 Detailed Description

```
template<typename _Tp>
struct std::remove_volatile< _Tp >
```

remove\_volatile

Definition at line 1338 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.1017 std::result\_of&lt; \_Signature &gt; Class Template Reference

## 5.1017.1 Detailed Description

```
template<typename _Signature>
class std::result_of< _Signature >
```

result\_of

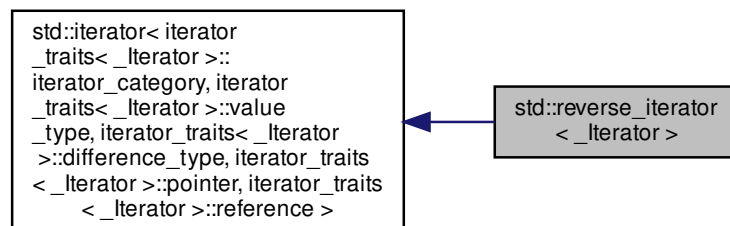
Definition at line 2074 of file type\_traits.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 5.1018 std::reverse\_iterator&lt; \_Iterator &gt; Class Template Reference

Inheritance diagram for std::reverse\_iterator< \_Iterator >:



## Public Types

- typedef \_\_traits\_type::difference\_type **difference\_type**
- typedef iterator\_traits<\_Iterator>::iterator\_category iterator\_category
- typedef \_Iterator **iterator\_type**
- typedef \_\_traits\_type::pointer **pointer**
- typedef \_\_traits\_type::reference **reference**
- typedef iterator\_traits<\_Iterator>::value\_type value\_type

## Public Member Functions

- \_GLIBCXX17\_CONSTEXPR reverse\_iterator ()
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator (iterator\_type \_\_x)
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator (const reverse\_iterator &\_\_x)
- template<typename \_Iter >  
\_GLIBCXX17\_CONSTEXPR reverse\_iterator (const reverse\_iterator<\_Iter> &\_\_x)
- \_GLIBCXX17\_CONSTEXPR iterator\_type base () const
- \_GLIBCXX17\_CONSTEXPR reference operator\* () const
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator operator+ (difference\_type \_\_n) const
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator & operator++ ()
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator operator++ (int)
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator & operator+= (difference\_type \_\_n)
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator operator- (difference\_type \_\_n) const
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator & operator-- ()
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator operator-- (int)
- \_GLIBCXX17\_CONSTEXPR reverse\_iterator & operator-= (difference\_type \_\_n)
- \_GLIBCXX17\_CONSTEXPR pointer operator-> () const
- \_GLIBCXX17\_CONSTEXPR reference operator[] (difference\_type \_\_n) const

## Protected Types

- typedef iterator\_traits<\_Iterator> \_\_traits\_type

## Protected Attributes

- \_Iterator **current**

### 5.1018.1 Detailed Description

```
template<typename _Iterator>
class std::reverse_iterator<_Iterator>
```

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator *i* is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array.* [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 101 of file bits/stl\_iterator.h.

## 5.1018.2 Member Typedef Documentation

### 5.1018.2.1 iterator\_category

```
typedef iterator_traits< _Iterator >::iterator_category std::iterator< iterator_traits< _Iterator >::iterator_category , iterator_traits< _Iterator >::value_type , iterator_traits< _Iterator >::difference_type , iterator_traits< _Iterator >::pointer , iterator_traits< _Iterator >::reference >::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

### 5.1018.2.2 value\_type

```
typedef iterator_traits< _Iterator >::value_type std::iterator< iterator_traits< _Iterator >::iterator_category , iterator_traits< _Iterator >::value_type , iterator_traits< _Iterator >::difference_type , iterator_traits< _Iterator >::pointer , iterator_traits< _Iterator >::reference >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

## 5.1018.3 Constructor & Destructor Documentation

### 5.1018.3.1 reverse\_iterator() [1/4]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR std::reverse_iterator< _Iterator >::reverse_iterator ( ) [inline]
```

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 127 of file `bits/stl_iterator.h`.

Referenced by `std::reverse_iterator< _Iterator >::operator+()`, and `std::reverse_iterator< _Iterator >::operator-()`.



**5.1018.3.2 reverse\_iterator()** [2/4]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR std::reverse_iterator< _Iterator >::reverse_iterator (
    iterator_type __x ) [inline], [explicit]
```

This iterator will move in the opposite direction that `x` does.

Definition at line 133 of file `bits/stl_iterator.h`.

**5.1018.3.3 reverse\_iterator()** [3/4]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR std::reverse_iterator< _Iterator >::reverse_iterator (
    const reverse_iterator< _Iterator > & __x ) [inline]
```

The copy constructor is normal.

Definition at line 139 of file `bits/stl_iterator.h`.

**5.1018.3.4 reverse\_iterator()** [4/4]

```
template<typename _Iterator>
template<typename _Iter >
_GLIBCXX17_CONSTEXPR std::reverse_iterator< _Iterator >::reverse_iterator (
    const reverse_iterator< _Iter > & __x ) [inline]
```

A `reverse_iterator` across other types can be copied if the underlying iterator can be converted to the type of `current`.

Definition at line 148 of file `bits/stl_iterator.h`.

**5.1018.4 Member Function Documentation****5.1018.4.1 base()**

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR iterator_type std::reverse_iterator< _Iterator >::base ( ) const [inline]
```

**Returns**

`current`, the iterator used for underlying work.

Definition at line 155 of file `bits/stl_iterator.h`.

Referenced by `std::operator==()`.

#### 5.1018.4.2 operator\*()

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reference std::reverse_iterator<_Iterator>::operator* ( ) const [inline]
```

##### Returns

A reference to the value at `-current`

This requires that `-current` is dereferenceable.

##### Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

Definition at line 169 of file `bits/stl_iterator.h`.

#### 5.1018.4.3 operator+()

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator std::reverse_iterator<_Iterator>::operator+ (
    difference_type __n ) const [inline]
```

##### Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 242 of file `bits/stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::reverse_iterator()`.

#### 5.1018.4.4 operator++() [1/2]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator& std::reverse_iterator<_Iterator>::operator++ ( ) [inline]
```

##### Returns

`*this`

Decrements the underlying iterator.

Definition at line 192 of file `bits/stl_iterator.h`.

**5.1018.4.5 operator++()** [2/2]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator std::reverse_iterator< _Iterator >::operator++ (
    int ) [inline]
```

**Returns**

The original value of `*this`

Decrements the underlying iterator.

Definition at line 204 of file `bits/stl_iterator.h`.

**5.1018.4.6 operator+=()**

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator& std::reverse_iterator< _Iterator >::operator+= (
    difference_type __n ) [inline]
```

**Returns**

`*this`

Moves the underlying iterator backwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 252 of file `bits/stl_iterator.h`.

**5.1018.4.7 operator-()**

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator std::reverse_iterator< _Iterator >::operator- (
    difference_type __n ) const [inline]
```

**Returns**

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 264 of file `bits/stl_iterator.h`.

References `std::reverse_iterator< _Iterator >::reverse_iterator()`.

**5.1018.4.8 operator--()** [1/2]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator& std::reverse_iterator< _Iterator >::operator-- ( ) [inline]
```

**Returns**

\*this

Increments the underlying iterator.

Definition at line 217 of file bits/stl\_iterator.h.

**5.1018.4.9 operator--()** [2/2]

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator std::reverse_iterator< _Iterator >::operator-- (
    int ) [inline]
```

**Returns**

A reverse\_iterator with the previous value of \*this

Increments the underlying iterator.

Definition at line 229 of file bits/stl\_iterator.h.

**5.1018.4.10 operator+=()**

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reverse_iterator& std::reverse_iterator< _Iterator >::operator+= (
    difference_type __n ) [inline]
```

**Returns**

\*this

Moves the underlying iterator forwards \_\_n steps. The underlying iterator must be a Random Access Iterator.

Definition at line 274 of file bits/stl\_iterator.h.

#### 5.1018.4.11 `operator->()`

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR pointer std::reverse_iterator< _Iterator >::operator-> ( ) const [inline]
```

##### Returns

A pointer to the value at `-current`

This requires that `-current` is dereferenceable.

Definition at line 183 of file `bits/stl_iterator.h`.

References `std::__addressof()`.

#### 5.1018.4.12 `operator[]()`

```
template<typename _Iterator>
_GLIBCXX17_CONSTEXPR reference std::reverse_iterator< _Iterator >::operator[] (
    difference_type __n ) const [inline]
```

##### Returns

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

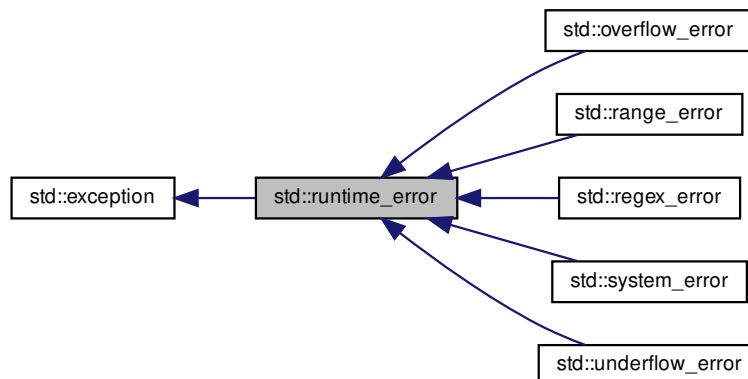
Definition at line 286 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

### 5.1019 `std::runtime_error` Class Reference

Inheritance diagram for `std::runtime_error`:



## Public Member Functions

- `runtime_error` (const [string](#) &\_\_arg) `_GLIBCXX_TXN_SAFE`
- `runtime_error` (const char \*) `_GLIBCXX_TXN_SAFE`
- virtual const char \* `what` () const `_GLIBCXX_TXN_SAFE_DYN` noexcept

### 5.1019.1 Detailed Description

One of two subclasses of exception.

Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 197 of file `stdexcept`.

### 5.1019.2 Constructor & Destructor Documentation

#### 5.1019.2.1 `runtime_error()`

```
std::runtime_error::runtime_error (
    const string & __arg ) [explicit]
```

Takes a character string describing the error.

### 5.1019.3 Member Function Documentation

#### 5.1019.3.1 `what()`

```
virtual const char* std::runtime_error::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.1020 `std::scoped_allocator_adaptor<_OuterAlloc, _InnerAllocs>` Class Template Reference

Inherits `_OuterAlloc`.

## Public Types

- typedef [\\_\\_traits::const\\_pointer](#) **const\_pointer**
- typedef [\\_\\_traits::const\\_void\\_pointer](#) **const\_void\_pointer**
- typedef [\\_\\_traits::difference\\_type](#) **difference\_type**
- typedef [\\_\\_inner\\_type::\\_\\_type](#) **inner\_allocator\_type**
- typedef [\\_\\_and\\_< typename \[\\\_\\\_traits::is\\\_always\\\_equal\]\(#\), typename \[allocator\\\_traits\]\(#\)< \\_InnerAllocs >::is\\_always\\_](#)↵  
[\\_equal... >::type](#) **is\_always\_equal**
- typedef [\\_OuterAlloc](#) **outer\_allocator\_type**
- typedef [\\_\\_traits::pointer](#) **pointer**
- typedef [\\_\\_or\\_< typename \[\\\_\\\_traits::propagate\\\_on\\\_container\\\_copy\\\_assignment\]\(#\), typename \[allocator\\\_traits\]\(#\)< \\_InnerAllocs >::propagate\\_on\\_container\\_copy\\_assignment... >::type](#) **propagate\_on\_container\_copy\_**↵  
**assignment**
- typedef [\\_\\_or\\_< typename \[\\\_\\\_traits::propagate\\\_on\\\_container\\\_move\\\_assignment\]\(#\), typename \[allocator\\\_traits\]\(#\)< \\_InnerAllocs >::propagate\\_on\\_container\\_move\\_assignment... >::type](#) **propagate\_on\_container\_move\_**↵  
**assignment**
- typedef [\\_\\_or\\_< typename \[\\\_\\\_traits::propagate\\\_on\\\_container\\\_swap\]\(#\), typename \[allocator\\\_traits\]\(#\)< \\_InnerAllocs >::propagate\\_on\\_container\\_swap... >::type](#) **propagate\_on\_container\_swap**
- typedef [\\_\\_traits::size\\_type](#) **size\_type**
- typedef [\\_\\_traits::value\\_type](#) **value\_type**
- typedef [\\_\\_traits::void\\_pointer](#) **void\_pointer**

## Public Member Functions

- template<typename [\\_Outer2](#) , typename = [\\_Constructible](#)<[\\_Outer2](#)>>>  
**scoped\_allocator\_adaptor** ([\\_Outer2](#) &&[\\_\\_outer](#), const [\\_InnerAllocs](#) &... [\\_\\_inner](#))
- **scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#) &[\\_\\_other](#))
- **scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#) &&[\\_\\_other](#))
- template<typename [\\_Outer2](#) , typename = [\\_Constructible](#)<const [\\_Outer2](#)>>>  
**scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#)< [\\_Outer2](#), [\\_InnerAllocs](#)... > &[\\_\\_other](#))
- template<typename [\\_Outer2](#) , typename = [\\_Constructible](#)<[\\_Outer2](#)>>>  
**scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#)< [\\_Outer2](#), [\\_InnerAllocs](#)... > &&[\\_\\_other](#))
- pointer **allocate** (size\_type [\\_\\_n](#))
- pointer **allocate** (size\_type [\\_\\_n](#), const\_void\_pointer [\\_\\_hint](#))
- template<typename [\\_Tp](#) , typename... [\\_Args](#)>  
void **construct** ([\\_Tp](#) \*[\\_\\_p](#), [\\_Args](#) &&... [\\_\\_args](#))
- template<typename [\\_T1](#) , typename [\\_T2](#) , typename... [\\_Args1](#), typename... [\\_Args2](#)>  
void **construct** ([pair](#)< [\\_T1](#), [\\_T2](#) > \*[\\_\\_p](#), [piecewise\\_construct\\_t](#), [tuple](#)< [\\_Args1](#)... > [\\_\\_x](#), [tuple](#)< [\\_Args2](#)... > [\\_\\_y](#))
- template<typename [\\_T1](#) , typename [\\_T2](#) >  
void **construct** ([pair](#)< [\\_T1](#), [\\_T2](#) > \*[\\_\\_p](#))
- template<typename [\\_T1](#) , typename [\\_T2](#) , typename [\\_Up](#) , typename [\\_Vp](#) >  
void **construct** ([pair](#)< [\\_T1](#), [\\_T2](#) > \*[\\_\\_p](#), [\\_Up](#) &&[\\_\\_u](#), [\\_Vp](#) &&[\\_\\_v](#))
- template<typename [\\_T1](#) , typename [\\_T2](#) , typename [\\_Up](#) , typename [\\_Vp](#) >  
void **construct** ([pair](#)< [\\_T1](#), [\\_T2](#) > \*[\\_\\_p](#), const [pair](#)< [\\_Up](#), [\\_Vp](#) > &[\\_\\_x](#))
- template<typename [\\_T1](#) , typename [\\_T2](#) , typename [\\_Up](#) , typename [\\_Vp](#) >  
void **construct** ([pair](#)< [\\_T1](#), [\\_T2](#) > \*[\\_\\_p](#), [pair](#)< [\\_Up](#), [\\_Vp](#) > &&[\\_\\_x](#))
- void **deallocate** (pointer [\\_\\_p](#), size\_type [\\_\\_n](#))
- template<typename [\\_Tp](#) >  
void **destroy** ([\\_Tp](#) \*[\\_\\_p](#))
- inner\_allocator\_type & **inner\_allocator** () noexcept

- `const inner_allocator_type & inner_allocator ()` const noexcept
- `size_type max_size ()` const
- `scoped_allocator_adaptor & operator= (const scoped_allocator_adaptor &)=default`
- `scoped_allocator_adaptor & operator= (scoped_allocator_adaptor &&)=default`
- `outer_allocator_type & outer_allocator ()` noexcept
- `const outer_allocator_type & outer_allocator ()` const noexcept
- `scoped_allocator_adaptor select_on_container_copy_construction ()` const

#### Friends

- `template<typename _Outer, typename... _Inner>`  
class **scoped\_allocator\_adaptor**
- `template<typename... >`  
class **\_\_inner\_type\_impl**
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool operator== (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b)` noexcept

#### 5.1020.1 Detailed Description

```
template<typename _OuterAlloc, typename... _InnerAllocs>
class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
```

Primary class template.

Definition at line 83 of file `scoped_allocator`.

The documentation for this class was generated from the following file:

- [scoped\\_allocator](#)

## 5.1021 std::seed\_seq Class Reference

### Public Types

- `typedef uint_least32_t result_type`

### Public Member Functions

- `seed_seq ()` noexcept
- `template<typename _IntType >`  
`seed_seq (std::initializer_list< _IntType > __il)`
- `template<typename _InputIterator >`  
`seed_seq (_InputIterator __begin, _InputIterator __end)`
- `seed_seq (const seed_seq &)=delete`
- `template<typename _RandomAccessIterator >`  
`void generate (_RandomAccessIterator __begin, _RandomAccessIterator __end)`
- `seed_seq & operator= (const seed_seq &)=delete`
- `template<typename _OutputIterator >`  
`void param (_OutputIterator __dest) const`
- `size_t size ()` const noexcept



#### 5.1021.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 5959 of file `random.h`.

#### 5.1021.2 Member Typedef Documentation

##### 5.1021.2.1 `result_type`

```
typedef uint_least32_t std::seed\_seq::result\_type
```

The type of the seed vales.

Definition at line 5963 of file `random.h`.

#### 5.1021.3 Constructor & Destructor Documentation

##### 5.1021.3.1 `seed_seq()`

```
std::seed\_seq::seed\_seq ( ) [inline], [noexcept]
```

Default constructor.

Definition at line 5966 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.1022 std::set&lt; \_Key, \_Compare, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Key [key\\_type](#)
  - typedef \_Key [value\\_type](#)
  - typedef \_Compare [key\\_compare](#)
  - typedef \_Compare [value\\_compare](#)
  - typedef \_Alloc [allocator\\_type](#)
- 
- typedef \_Alloc\_traits::pointer [pointer](#)
  - typedef \_Alloc\_traits::const\_pointer [const\\_pointer](#)
  - typedef \_Alloc\_traits::reference [reference](#)
  - typedef \_Alloc\_traits::const\_reference [const\\_reference](#)
  - typedef \_Rep\_type::const\_iterator [iterator](#)
  - typedef \_Rep\_type::const\_iterator [const\\_iterator](#)
  - typedef \_Rep\_type::const\_reverse\_iterator [reverse\\_iterator](#)
  - typedef \_Rep\_type::const\_reverse\_iterator [const\\_reverse\\_iterator](#)
  - typedef \_Rep\_type::size\_type [size\\_type](#)
  - typedef \_Rep\_type::difference\_type [difference\\_type](#)

## Public Member Functions

- [set](#) ()=default
- [set](#) (const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator >  
  [set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator >  
  [set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [set](#) &)=default
- [set](#) ([set](#) &&)=default
- [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Compare &\_\_comp=\_Compare(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [allocator\\_type](#) &\_\_a)
- [set](#) (const [set](#) &\_\_x, const [allocator\\_type](#) &\_\_a)
- [set](#) ([set](#) &&\_\_x, const [allocator\\_type](#) &\_\_a) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value &&\_\_a.\_Alloc\_traits::\_S\_always\_equal())
- [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
  [set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)
- [~set](#) ()=default
- [iterator begin](#) () const noexcept
- [iterator cbegin](#) () const noexcept

- `iterator cend` () const noexcept
  - void `clear` () noexcept
  - `reverse_iterator crbegin` () const noexcept
  - `reverse_iterator crend` () const noexcept
  - template<typename... \_Args>  
`std::pair< iterator, bool > emplace` (\_Args &&... \_\_args)
  - template<typename... \_Args>  
`iterator emplace_hint` (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool `empty` () const noexcept
  - `iterator end` () const noexcept
  - \_GLIBCXX\_ABI\_TAG\_CXX11 `iterator erase` (const\_iterator \_\_position)
  - `size_type erase` (const key\_type &\_\_x)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 `iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
  - `allocator_type get_allocator` () const noexcept
  - `std::pair< iterator, bool > insert` (const value\_type &\_\_x)
  - `std::pair< iterator, bool > insert` (value\_type &&\_\_x)
  - `iterator insert` (const\_iterator \_\_position, const value\_type &\_\_x)
  - `iterator insert` (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void `insert` (initializer\_list< value\_type > \_\_l)
  - `key_compare key_comp` () const
  - `size_type max_size` () const noexcept
  - `set & operator=` (const set &)=default
  - `set & operator=` (set &&)=default
  - `set & operator=` (initializer\_list< value\_type > \_\_l)
  - `reverse_iterator rbegin` () const noexcept
  - `reverse_iterator rend` () const noexcept
  - `size_type size` () const noexcept
  - void `swap` (set &\_\_x) noexcept(\*conditional \*)
  - `value_compare value_comp` () const
- 
- `size_type count` (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto `count` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- 
- `iterator find` (const key\_type &\_\_x)
  - `const_iterator find` (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto `find` (const \_Kt &\_\_x) -> decltype(iterator
  - template<typename \_Kt >  
auto `find` (const \_Kt &\_\_x) const -> decltype(const\_iterator

- [iterator lower\\_bound](#) (const [key\\_type](#) &\_\_x)
  - [const\\_iterator lower\\_bound](#) (const [key\\_type](#) &\_\_x) const
  - template<typename \_Kt >  
auto [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype([iterator](#)(\_M.t.\_M\_lower\_bound\_tr(\_\_x)))
  - template<typename \_Kt >  
auto [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype([const\\_iterator](#)(\_M.t.\_M\_lower\_bound\_tr(\_\_x)))
- 
- [iterator upper\\_bound](#) (const [key\\_type](#) &\_\_x)
  - [const\\_iterator upper\\_bound](#) (const [key\\_type](#) &\_\_x) const
  - template<typename \_Kt >  
auto [upper\\_bound](#) (const \_Kt &\_\_x) -> decltype([iterator](#)(\_M.t.\_M\_upper\_bound\_tr(\_\_x)))
  - template<typename \_Kt >  
auto [upper\\_bound](#) (const \_Kt &\_\_x) const -> decltype([iterator](#)(\_M.t.\_M\_upper\_bound\_tr(\_\_x)))
- 
- [std::pair< iterator, iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x)
  - [std::pair< const\\_iterator, const\\_iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x) const
  - template<typename \_Kt >  
auto [equal\\_range](#) (const \_Kt &\_\_x) -> decltype([pair](#)< [iterator](#), [iterator](#) >(\_M.t.\_M\_equal\_range\_tr(\_\_x)))
  - template<typename \_Kt >  
auto [equal\\_range](#) (const \_Kt &\_\_x) const -> decltype([pair](#)< [iterator](#), [iterator](#) >(\_M.t.\_M\_equal\_range\_tr(\_\_x)))

## Friends

- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator**< (const [set](#)< \_K1, \_C1, \_A1 > &, const [set](#)< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1 >  
bool **operator**== (const [set](#)< \_K1, \_C1, \_A1 > &, const [set](#)< \_K1, \_C1, \_A1 > &)

### 5.1022.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::set< _Key, _Compare, _Alloc >
```

A standard container made up of unique keys, which can be retrieved in logarithmic time.

#### Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

Definition at line 70 of file `stl_multiset.h`.

## 5.1022.2 Member Typedef Documentation

### 5.1022.2.1 `allocator_type`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc std::set< _Key, _Compare, _Alloc >::allocator\_type
```

Public typedefs.

Definition at line 124 of file `stl_set.h`.

### 5.1022.2.2 `const_iterator`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::const\_iterator
```

Iterator-related typedefs.

Definition at line 148 of file `stl_set.h`.

### 5.1022.2.3 `const_pointer`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::const_pointer std::set< _Key, _Compare, _Alloc >::const\_pointer
```

Iterator-related typedefs.

Definition at line 141 of file `stl_set.h`.

#### 5.1022.2.4 const\_reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
typedef _Alloc_traits::const_reference std::set< _Key, _Compare, _Alloc >::const_reference
```

Iterator-related typedefs.

Definition at line 143 of file stl\_set.h.

#### 5.1022.2.5 const\_reverse\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_iterator
```

Iterator-related typedefs.

Definition at line 150 of file stl\_set.h.

#### 5.1022.2.6 difference\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
typedef _Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type
```

Iterator-related typedefs.

Definition at line 152 of file stl\_set.h.

#### 5.1022.2.7 iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 147 of file stl\_set.h.

#### 5.1022.2.8 key\_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare
```

Public typedefs.

Definition at line 122 of file stl\_set.h.

#### 5.1022.2.9 key\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::key_type
```

Public typedefs.

Definition at line 109 of file stl\_set.h.

#### 5.1022.2.10 pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::pointer std::set< _Key, _Compare, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 140 of file stl\_set.h.

#### 5.1022.2.11 reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::reference std::set< _Key, _Compare, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 142 of file stl\_set.h.

#### 5.1022.2.12 reverse\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator
```

Iterator-related typedefs.

Definition at line 149 of file stl\_set.h.

#### 5.1022.2.13 size\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Rep_type::size_type std::set< _Key, _Compare, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 151 of file stl\_set.h.

#### 5.1022.2.14 value\_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::value_compare
```

Public typedefs.

Definition at line 123 of file stl\_set.h.

#### 5.1022.2.15 value\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::value_type
```

Public typedefs.

Definition at line 121 of file stl\_set.h.

### 5.1022.3 Constructor & Destructor Documentation



**5.1022.3.1** `set()` [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set ( ) [default]
```

Default constructor creates no elements.

**5.1022.3.2** `set()` [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a set with no elements.

**Parameters**

<code>__comp</code>	Comparator to use.
<code>__a</code>	An allocator object.

Definition at line 176 of file `stl_set.h`.

**5.1022.3.3** `set()` [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::set<_Key, _Compare, _Alloc >::set (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Builds a set from a range.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 191 of file `stl_set.h`.

## 5.1022.3.4 set() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::set< _Key, _Compare, _Alloc >::set (
    _InputIterator __first,
    _InputIterator __last,
    const _Compare & __comp,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a set from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a set consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 208 of file `stl_set.h`.

## 5.1022.3.5 set() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
    const set< _Key, _Compare, _Alloc > & ) [default]
```

Set copy constructor.

Whether the allocator is copied depends on the allocator traits.

## 5.1022.3.6 set() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
    set< _Key, _Compare, _Alloc > && ) [default]
```

Set move constructor

The newly-created set contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, set.

**5.1022.3.7 set()** [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    initializer_list< value_type > __l,
    const _Compare & __comp = _Compare(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a set from an `initializer_list`.

**Parameters**

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in  $N$  if the list is already sorted, and  $N\log N$  otherwise (where  $N$  is `__l.size()`).

Definition at line 243 of file `stl_set.h`.

**5.1022.3.8 set()** [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    const allocator_type & __a ) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 251 of file `stl_set.h`.

**5.1022.3.9 set()** [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc >::set (
    const set<_Key, _Compare, _Alloc > & __x,
    const allocator_type & __a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 255 of file `stl_set.h`.

## 5.1022.3.10 set() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
    set< _Key, _Compare, _Alloc > && __x,
    const allocator_type & __a ) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 259 of file stl\_set.h.

## 5.1022.3.11 set() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
    initializer_list< value_type > __l,
    const allocator_type & __a ) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 265 of file stl\_set.h.

## 5.1022.3.12 set() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::set< _Key, _Compare, _Alloc >::set (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a ) [inline]
```

Allocator-extended range constructor.

Definition at line 271 of file stl\_set.h.

## 5.1022.3.13 ~set()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::~set ( ) [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### 5.1022.4 Member Function Documentation

##### 5.1022.4.1 begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 344 of file `stl_set.h`.

##### 5.1022.4.2 cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 381 of file `stl_set.h`.

##### 5.1022.4.3 cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 390 of file `stl_set.h`.

##### 5.1022.4.4 clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 733 of file `stl_set.h`.

##### 5.1022.4.5 count() <sup>[1/2]</sup>

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::set< _Key, _Compare, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements.

## Parameters

$\_x$	Element to located.
-------	---------------------

## Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 748 of file stl\_set.h.

## 5.1022.4.6 count() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::count (
    const _Kt & __x ) const -> decltype(_M_t._M_count_tr(__x))    [inline]
```

Finds the number of elements.

## Parameters

$\_x$	Element to located.
-------	---------------------

## Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 754 of file stl\_set.h.

## 5.1022.4.7 crbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crbegin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 399 of file stl\_set.h.

#### 5.1022.4.8 `crend()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 408 of file `stl_set.h`.

#### 5.1022.4.9 `emplace()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Attempts to build and insert an element into the set.

##### Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

##### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 462 of file `stl_set.h`.

#### 5.1022.4.10 `emplace_hint()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
iterator std::set< _Key, _Compare, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Attempts to insert an element into the set.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

## Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 488 of file `stl_set.h`.

5.1022.4.11 `empty()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::set<_Key, _Compare, _Alloc>::empty ( ) const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 414 of file `stl_set.h`.

5.1022.4.12 `end()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc>::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 353 of file `stl_set.h`.

5.1022.4.13 `equal_range()` [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<iterator, iterator> std::set<_Key, _Compare, _Alloc>::equal_range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.



**Parameters**

<code>_↔</code>	Key to be located.
<code>_X</code>	

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 879 of file `stl_set.h`.

**5.1022.4.14 equal\_range()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
_Key>>  
std::pair<const_iterator, const_iterator> std::set< _Key, _Compare, _Alloc >::equal_range (  
    const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>_↔</code>	Key to be located.
<code>_X</code>	

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 883 of file `stl_set.h`.

## 5.1022.4.15 equal\_range() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

## Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 889 of file stl\_set.h.

## 5.1022.4.16 equal\_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::equal_range (
    const _Kt & __x ) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_↔
tr(__x))) [inline]
```

Finds a subsequence matching given key.

## Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 895 of file `stl_set.h`.

**5.1022.4.17 erase()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (  
    const_iterator __position ) [inline]
```

Erases an element from a set.

**Parameters**

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 654 of file `stl_set.h`.

**5.1022.4.18 erase()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵  
_Key>>  
size_type std::set< _Key, _Compare, _Alloc >::erase (  
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

## Parameters

<code>_↵</code>	Key of element to be erased.
<code>_X</code>	

## Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 684 of file stl\_set.h.

## 5.1022.4.19 erase() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from a set.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

## Returns

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 706 of file stl\_set.h.

## 5.1022.4.20 find() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in a set.

**Parameters**

<code>_↵</code>	Element to be located.
<code>_X</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 775 of file `stl_set.h`.

**5.1022.4.21 find()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in a set.

**Parameters**

<code>_↵</code>	Element to be located.
<code>_X</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 779 of file `stl_set.h`.

**5.1022.4.22 find()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::find (
    const _Kt & __x ) -> decltype(iterator [inline]
```

Tries to locate an element in a set.

## Parameters

<code>_↵</code>	Element to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 785 of file `stl_set.h`.

## 5.1022.4.23 find() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::find (
    const _Kt & __x ) const -> decltype(const_iterator [inline])
```

Tries to locate an element in a set.

## Parameters

<code>_↵</code>	Element to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 791 of file `stl_set.h`.

## 5.1022.4.24 get\_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

Definition at line 335 of file `stl_set.h`.

**5.1022.4.25** `insert()` [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<iterator, bool> std::set<_Key, _Compare, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Attempts to insert an element into the set.

**Parameters**

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 509 of file `stl_set.h`.

References `std::pair<_T1, _T2>::first`, and `std::pair<_T1, _T2>::second`.

Referenced by `std::set<_Key, _Compare, _Alloc>::insert()`.

**5.1022.4.26** `insert()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::insert (
    const_iterator __position,
    const value_type & __x ) [inline]
```

Attempts to insert an element into the set.

**Parameters**

<code>__position</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 546 of file `stl_set.h`.

**5.1022.4.27 insert()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
void std::set< _Key, _Compare, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 566 of file `stl_set.h`.

**5.1022.4.28 insert()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set< _Key, _Compare, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the set.



**Parameters**

<code>↵</code>	A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted.
<code>_↵</code>	
<code>↵</code>	
<code>_↵</code>	
<code>/</code>	

Complexity similar to that of the range constructor.

Definition at line 578 of file `stl_set.h`.

References `std::set<_Key, _Compare, _Alloc >::insert()`.

**5.1022.4.29 `key_comp()`**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
key_compare std::set<_Key, _Compare, _Alloc >::key_comp ( ) const [inline]
```

Returns the comparison object with which the set was constructed.

Definition at line 327 of file `stl_set.h`.

**5.1022.4.30 `lower_bound()`** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set<_Key, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>_↵</code>	Key to be located.
<code>_X</code>	

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 810 of file `stl_set.h`.

## 5.1022.4.31 lower\_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
    const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 814 of file stl\_set.h.

## 5.1022.4.32 lower\_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 820 of file stl\_set.h.

**5.1022.4.33** `lower_bound()` [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
    const _Kt & __x ) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

<code>_Key</code>	Key to be located.
<code>__x</code>	

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 826 of file `stl_set.h`.

**5.1022.4.34** `max_size()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::set< _Key, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 424 of file `stl_set.h`.

**5.1022.4.35** `operator=()` [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
set& std::set< _Key, _Compare, _Alloc >::operator= (
    const set< _Key, _Compare, _Alloc > & ) [default]
```

Set assignment operator.

Whether the allocator is copied depends on the allocator traits.

**5.1022.4.36 operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
set& std::set< _Key, _Compare, _Alloc >::operator= (
    set< _Key, _Compare, _Alloc > && ) [default]
```

Move assignment operator.

**5.1022.4.37 operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
set& std::set< _Key, _Compare, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Set list assignment operator.

**Parameters**

↩	An initializer_list.
↩	
↩	
↩	
/	

This function fills a set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned.

Definition at line 316 of file `stl_set.h`.

**5.1022.4.38 rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 362 of file `stl_set.h`.

**5.1022.4.39** `rend()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 371 of file `stl_set.h`.

**5.1022.4.40** `size()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the size of the set.

Definition at line 419 of file `stl_set.h`.

**5.1022.4.41** `swap()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
void std::set< _Key, _Compare, _Alloc >::swap (
    set< _Key, _Compare, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another set.

**Parameters**

<code>__x</code>	A set of the same element and allocator types.
------------------	--

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 441 of file `stl_set.h`.

## 5.1022.4.42 upper\_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>_↵</code>	Key to be located.
<code>_X</code>	

## Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 840 of file stl\_set.h.

## 5.1022.4.43 upper\_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

## Parameters

<code>_↵</code>	Key to be located.
<code>_X</code>	

## Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 844 of file stl\_set.h.

## 5.1022.4.44 upper\_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::upper_bound (
    const key_type & __x ) [inline]
```

```
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

$\_x$	Key to be located.
-------	--------------------

#### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 850 of file stl\_set.h.

#### 5.1022.4.45 upper\_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
    const _Kt & __x ) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))    [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

$\_x$	Key to be located.
-------	--------------------

#### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 856 of file stl\_set.h.

#### 5.1022.4.46 value\_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::set< _Key, _Compare, _Alloc >::value_comp ( ) const    [inline]
```

Returns the comparison object with which the set was constructed.

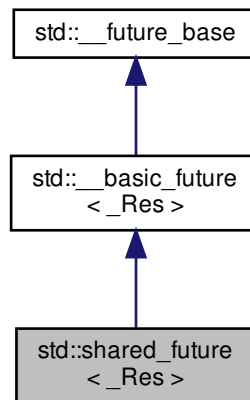
Definition at line 331 of file stl\_set.h.

The documentation for this class was generated from the following files:

- [stl\\_multiset.h](#)
- [stl\\_set.h](#)

## 5.1023 `std::shared_future<_Res>` Class Template Reference

Inheritance diagram for `std::shared_future<_Res>`:



### Public Types

- `template<typename _Res>`  
`using _Ptr = unique\_ptr<_Res, _Result_base::_Deleter>`
- `using _State_base = _State_baseV2`

### Public Member Functions

- `shared\_future (const shared\_future &__sf) noexcept`
- `shared\_future (future<_Res> &&__uf) noexcept`
- `shared\_future (shared\_future &&__sf) noexcept`
- `const _Res & get () const`
- `shared\_future & operator= (const shared\_future &__sf) noexcept`
- `shared\_future & operator= (shared\_future &&__sf) noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `template<typename _Rep, typename _Period>`  
`future\_status wait\_for (const chrono::duration<_Rep, _Period> &__rel) const`
- `template<typename _Clock, typename _Duration>`  
`future\_status wait\_until (const chrono::time\_point<_Clock, _Duration> &__abs) const`



### Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static _Ptr<_Result_alloc<_Res, _Allocator > > _S_allocate_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static _Ptr<_Result<_Res > > _S_allocate_result (const std::allocator<_Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared_ptr<_State_base > _S_make_async_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared_ptr<_State_base > _S_make_deferred_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static _Task_setter<_Res_ptr, _BoundFn > _S_task_setter (_Res_ptr &__ptr, _BoundFn &__call)`

### Protected Types

- `typedef __future_base::_Result<_Res > & __result_type`
- `typedef shared_ptr<_State_base > __state_type`

### Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

#### 5.1023.1 Detailed Description

```
template<typename _Res>
class std::shared_future<_Res >
```

Primary template for shared\_future.

Definition at line 128 of file future.

#### 5.1023.2 Member Typedef Documentation

##### 5.1023.2.1 \_Ptr

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A unique\_ptr for result objects.

Definition at line 223 of file future.

### 5.1023.3 Constructor & Destructor Documentation

#### 5.1023.3.1 shared\_future() [1/3]

```
template<typename _Res >
std::shared_future<_Res>::shared_future (
    const shared_future<_Res> & __sf ) [inline], [noexcept]
```

Copy constructor.

Definition at line 899 of file future.

#### 5.1023.3.2 shared\_future() [2/3]

```
template<typename _Res >
std::shared_future<_Res>::shared_future (
    future<_Res> && __uf ) [inline], [noexcept]
```

Construct from a future rvalue.

Definition at line 902 of file future.

#### 5.1023.3.3 shared\_future() [3/3]

```
template<typename _Res >
std::shared_future<_Res>::shared_future (
    shared_future<_Res> && __sf ) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

Definition at line 907 of file future.

### 5.1023.4 Member Function Documentation

#### 5.1023.4.1 \_M\_get\_result()

```
template<typename _Res>
__result_type std::__basic_future<_Res>::_M_get_result ( ) const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 714 of file future.

#### 5.1023.4.2 `get()`

```
template<typename _Res >  
const _Res& std::shared\_future< _Res >::get ( ) const [inline]
```

Retrieving the value.

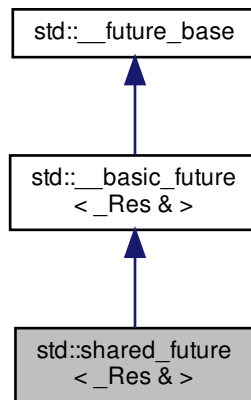
Definition at line 925 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

#### 5.1024 `std::shared_future< _Res & >` Class Template Reference

Inheritance diagram for `std::shared_future< _Res & >`:



#### Public Types

- `template<typename _Res >`  
  `using _Ptr = unique\_ptr< _Res, _Result_base::_Deleter >`
- `using _State_base = _State_baseV2`

## Public Member Functions

- [shared\\_future](#) (const [shared\\_future](#) &\_\_sf)
- [shared\\_future](#) (future<\_Res & > &&\_\_uf) noexcept
- [shared\\_future](#) ([shared\\_future](#) &&\_\_sf) noexcept
- [\\_Res](#) & [get](#) () const
- [shared\\_future](#) & [operator=](#) (const [shared\\_future](#) &\_\_sf)
- [shared\\_future](#) & [operator=](#) ([shared\\_future](#) &&\_\_sf) noexcept
- bool [valid](#) () const noexcept
- void [wait](#) () const
- [future\\_status](#) [wait\\_for](#) (const [chrono::duration](#)<\_Rep, \_Period > &\_\_rel) const
- [future\\_status](#) [wait\\_until](#) (const [chrono::time\\_point](#)<\_Clock, \_Duration > &\_\_abs) const

## Static Public Member Functions

- template<typename \_Res, typename \_Allocator >  
static [\\_Ptr](#)<[\\_Result\\_alloc](#)<\_Res, \_Allocator > > [\\_S\\_allocate\\_result](#) (const \_Allocator &\_\_a)
- template<typename \_Res, typename \_Tp >  
static [\\_Ptr](#)<[\\_Result](#)<\_Res > > [\\_S\\_allocate\\_result](#) (const [std::allocator](#)<\_Tp > &\_\_a)
- template<typename \_BoundFn >  
static [std::shared\\_ptr](#)<\_State\_base > [\\_S\\_make\\_async\\_state](#) (\_BoundFn &&\_\_fn)
- template<typename \_BoundFn >  
static [std::shared\\_ptr](#)<\_State\_base > [\\_S\\_make\\_deferred\\_state](#) (\_BoundFn &&\_\_fn)
- template<typename \_Res\_ptr, typename \_BoundFn >  
static [\\_Task\\_setter](#)<\_Res\_ptr, \_BoundFn > [\\_S\\_task\\_setter](#) (\_Res\_ptr &\_\_ptr, \_BoundFn &\_\_call)

## Protected Types

- typedef [\\_\\_future\\_base::Result](#)<\_Res & > & [\\_\\_result\\_type](#)
- typedef [shared\\_ptr](#)<\_State\_base > [\\_\\_state\\_type](#)

## Protected Member Functions

- [\\_\\_result\\_type](#) [\\_M\\_get\\_result](#) () const
- void [\\_M\\_swap](#) ([\\_\\_basic\\_future](#) &\_\_that) noexcept

## 5.1024.1 Detailed Description

```
template<typename _Res>
class std::shared_future<_Res & >
```

Partial specialization for [shared\\_future](#)<R&>

Definition at line 930 of file future.

## 5.1024.2 Member Typedef Documentation

### 5.1024.2.1 \_Ptr

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A unique\_ptr for result objects.

Definition at line 223 of file future.

## 5.1024.3 Constructor & Destructor Documentation

### 5.1024.3.1 shared\_future() [1/3]

```
template<typename _Res >
std::shared_future< _Res & >::shared_future (
    const shared_future< _Res & > & __sf ) [inline]
```

Copy constructor.

Definition at line 938 of file future.

### 5.1024.3.2 shared\_future() [2/3]

```
template<typename _Res >
std::shared_future< _Res & >::shared_future (
    future< _Res & > && __uf ) [inline], [noexcept]
```

Construct from a future rvalue.

Definition at line 941 of file future.

### 5.1024.3.3 shared\_future() [3/3]

```
template<typename _Res >
std::shared_future< _Res & >::shared_future (
    shared_future< _Res & > && __sf ) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

Definition at line 946 of file future.

## 5.1024.4 Member Function Documentation

5.1024.4.1 `_M_get_result()`

```
__result_type std::__basic_future< _Res & >::_M_get_result ( ) const [inline], [protected],  
[inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 714 of file future.

5.1024.4.2 `get()`

```
template<typename _Res >  
_Res& std::shared_future< _Res & >::get ( ) const [inline]
```

Retrieving the value.

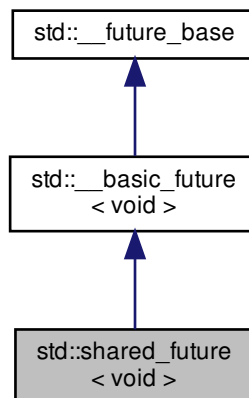
Definition at line 964 of file future.

The documentation for this class was generated from the following file:

- [future](#)

5.1025 `std::shared_future< void >` Class Template Reference

Inheritance diagram for `std::shared_future< void >`:



## Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr< _Res, Result\_base::Deleter >`
- `using \_State\_base = \_State\_baseV2`

## Public Member Functions

- `shared\_future (const shared\_future &__sf)`
- `shared\_future (future< void > &&__uf) noexcept`
- `shared\_future (shared\_future &&__sf) noexcept`
- `void get () const`
- `shared\_future & operator= (const shared\_future &__sf)`
- `shared\_future & operator= (shared\_future &&__sf) noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `future\_status wait\_for (const chrono::duration< _Rep, _Period > &__rel) const`
- `future\_status wait\_until (const chrono::time\_point< _Clock, _Duration > &__abs) const`

## Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static _Ptr< \_Result\_alloc< _Res, _Allocator > > \_S\_allocate\_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static _Ptr< \_Result< _Res > > \_S\_allocate\_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared\_ptr< \_State\_base > \_S\_make\_async\_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared\_ptr< \_State\_base > \_S\_make\_deferred\_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static \_Task\_setter< _Res_ptr, _BoundFn > \_S\_task\_setter (_Res_ptr &__ptr, _BoundFn &__call)`

## Protected Types

- `typedef \_future\_base::Result< void > & \_\_result\_type`
- `typedef shared\_ptr< \_State\_base > \_\_state\_type`

## Protected Member Functions

- `\_\_result\_type \_M\_get\_result () const`
- `void \_M\_swap (\_\_basic\_future &__that) noexcept`

## 5.1025.1 Detailed Description

```
template<>
class std::shared\_future< void >
```

Explicit specialization for `shared_future<void>`

Definition at line 969 of file `future`.

## 5.1025.2 Member Typedef Documentation

### 5.1025.2.1 \_Ptr

```
template<typename _Res >  
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A unique\_ptr for result objects.

Definition at line 223 of file future.

## 5.1025.3 Constructor & Destructor Documentation

### 5.1025.3.1 shared\_future() [1/3]

```
std::shared_future< void >::shared_future (  
    const shared_future< void > & __sf ) [inline]
```

Copy constructor.

Definition at line 977 of file future.

### 5.1025.3.2 shared\_future() [2/3]

```
std::shared_future< void >::shared_future (  
    future< void > && __uf ) [inline], [noexcept]
```

Construct from a future rvalue.

Definition at line 980 of file future.

### 5.1025.3.3 shared\_future() [3/3]

```
std::shared_future< void >::shared_future (  
    shared_future< void > && __sf ) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

Definition at line 985 of file future.



#### 5.1025.4 Member Function Documentation

##### 5.1025.4.1 `_M_get_result()`

```
__result_type std::__basic_future< void >::_M_get_result ( ) const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 714 of file future.

The documentation for this class was generated from the following file:

- [future](#)

#### 5.1026 `std::shared_lock< _Mutex >` Class Template Reference

##### Public Types

- typedef `_Mutex` **mutex\_type**

##### Public Member Functions

- **shared\_lock** (mutex\_type &\_\_m)
- **shared\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **shared\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **shared\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#))
- template<typename \_Clock, typename \_Duration >  
**shared\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- template<typename \_Rep, typename \_Period >  
**shared\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel\_time)
- **shared\_lock** ([shared\\_lock](#) const &)=delete
- **shared\_lock** ([shared\\_lock](#) &&\_\_sl) noexcept
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- [shared\\_lock](#) & **operator=** ([shared\\_lock](#) const &)=delete
- [shared\\_lock](#) & **operator=** ([shared\\_lock](#) &&\_\_sl) noexcept
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** ([shared\\_lock](#) &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel\_time)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- void **unlock** ()

## 5.1026.1 Detailed Description

```
template<typename _Mutex>
class std::shared_lock< _Mutex >
```

shared\_lock

Definition at line 541 of file shared\_mutex.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 5.1027 std::shared\_ptr&lt; \_Tp &gt; Class Template Reference

Inherits std::\_\_shared\_ptr< \_Tp, \_Lp >.

## Public Types

- using **element\_type** = typename \_\_shared\_ptr< \_Tp >::element\_type

## Public Member Functions

- constexpr [shared\\_ptr](#) () noexcept
- **shared\_ptr** (const [shared\\_ptr](#) &) noexcept=default
- template<typename \_Yp, typename = \_Constructible<\_Yp\*>>  
[shared\\_ptr](#) (\_Yp \*\_\_p)
- template<typename \_Yp, typename \_Deleter, typename = \_Constructible<\_Yp\*, \_Deleter>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d)
- template<typename \_Deleter >  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d)
- template<typename \_Yp, typename \_Deleter, typename \_Alloc, typename = \_Constructible<\_Yp\*, \_Deleter, \_Alloc>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- template<typename \_Deleter, typename \_Alloc >  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- template<typename \_Yp >  
[shared\\_ptr](#) (const [shared\\_ptr](#)< \_Yp > &\_\_r, element\_type \*\_\_p) noexcept
- template<typename \_Yp, typename = \_Constructible<const shared\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const [shared\\_ptr](#)< \_Yp > &\_\_r) noexcept
- [shared\\_ptr](#) ([shared\\_ptr](#) &&\_\_r) noexcept
- template<typename \_Yp, typename = \_Constructible<shared\_ptr<\_Yp>>>  
[shared\\_ptr](#) ([shared\\_ptr](#)< \_Yp > &&\_\_r) noexcept
- template<typename \_Yp, typename = \_Constructible<const weak\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const [weak\\_ptr](#)< \_Yp > &\_\_r)
- template<typename \_Yp, typename \_Del, typename = \_Constructible<unique\_ptr<\_Yp, \_Del>>>  
**shared\_ptr** ([unique\\_ptr](#)< \_Yp, \_Del > &&\_\_r)
- constexpr [shared\\_ptr](#) (nullptr\_t) noexcept

- `template<typename _Tp1, typename >`  
`shared_ptr (std::auto_ptr< _Tp1 > &&__r)`
- `element_type * get () const noexcept`
- `operator bool () const`
- `element_type & operator* () const noexcept`
- `element_type * operator-> () const noexcept`
- `shared_ptr & operator= (const shared_ptr &) noexcept=default`
- `template<typename _Yp >`  
`_Assignable< const shared_ptr< _Yp > & > operator= (const shared_ptr< _Yp > &__r) noexcept`
- `shared_ptr & operator= (shared_ptr &&__r) noexcept`
- `template<class _Yp >`  
`_Assignable< shared_ptr< _Yp > > operator= (shared_ptr< _Yp > &&__r) noexcept`
- `template<typename _Yp, typename _Del >`  
`_Assignable< unique_ptr< _Yp, _Del > > operator= (unique_ptr< _Yp, _Del > &&__r)`
- `template<typename _Tp1 >`  
`bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs) const noexcept`
- `template<typename _Tp1 >`  
`bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs) const noexcept`
- `void reset () noexcept`
- `template<typename _Yp >`  
`_SafeConv< _Yp > reset (_Yp *__p)`
- `template<typename _Yp, typename _Deleter >`  
`_SafeConv< _Yp > reset (_Yp *__p, _Deleter __d)`
- `template<typename _Yp, typename _Deleter, typename _Alloc >`  
`_SafeConv< _Yp > reset (_Yp *__p, _Deleter __d, _Alloc __a)`
- `void swap (__shared_ptr< _Tp, _Lp > &__other) noexcept`
- `bool unique () const noexcept`
- `long use_count () const noexcept`

## Friends

- `template<typename _Yp, typename _Alloc, typename... _Args>`  
`shared_ptr< _Yp > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `class weak_ptr< _Tp >`

### 5.1027.1 Detailed Description

```
template<typename _Tp>
class std::shared_ptr< _Tp >
```

A smart pointer with reference-counted copy semantics.

The object pointed to is deleted when the last `shared_ptr` pointing to it is destroyed or reset.

Definition at line 103 of file `bits/shared_ptr.h`.

### 5.1027.2 Constructor & Destructor Documentation

## 5.1027.2.1 shared\_ptr() [1/12]

```
template<typename _Tp>
constexpr std::shared_ptr< _Tp >::shared_ptr ( ) [inline], [noexcept]
```

Construct an empty shared\_ptr.

## Postcondition

use\_count()==0 && get()==0

Definition at line 127 of file bits/shared\_ptr.h.

## 5.1027.2.2 shared\_ptr() [2/12]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<_Yp*>>
std::shared_ptr< _Tp >::shared_ptr (
    _Yp * __p ) [inline], [explicit]
```

Construct a shared\_ptr that owns the pointer \_\_p.

## Parameters

$\leftarrow$ __p	A pointer that is convertible to element_type*.
---------------------	---

## Postcondition

use\_count() == 1 && get() == \_\_p

## Exceptions

std::bad_alloc, in	which case delete __p is called.
--------------------	----------------------------------

Definition at line 139 of file bits/shared\_ptr.h.

## 5.1027.2.3 shared\_ptr() [3/12]

```
template<typename _Tp>
template<typename _Yp , typename _Deleter , typename = _Constructible<_Yp*, _Deleter>>
std::shared_ptr< _Tp >::shared_ptr (
    _Yp * __p,
    _Deleter __d ) [inline]
```

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.

**Parameters**

$\_p$	A pointer.
$\_d$	A deleter.

**Postcondition**

`use_count() == 1 && get() ==  $\_p$`

**Exceptions**

<i>std::bad_alloc</i> , in	which case <code><math>\_d(\_p)</math></code> is called.
----------------------------	--

Requirements: `_Deleter`'s copy constructor and destructor must not throw

`__shared_ptr` will release  `$\_p$`  by calling  `$\_d(\_p)$`

Definition at line 156 of file `bits/shared_ptr.h`.

**5.1027.2.4 `shared_ptr()`** [4/12]

```
template<typename _Tp>
template<typename _Deleter >
std::shared_ptr< _Tp >::shared_ptr (
    nullptr_t  $\_p$ ,
    _Deleter  $\_d$  ) [inline]
```

Construct a `shared_ptr` that owns a null pointer and the deleter  `$\_d$` .

**Parameters**

$\_p$	A null pointer constant.
$\_d$	A deleter.

**Postcondition**

`use_count() == 1 && get() ==  $\_p$`

**Exceptions**

<i>std::bad_alloc</i> , in	which case <code><math>\_d(\_p)</math></code> is called.
----------------------------	--

Requirements: \_Deleter's copy constructor and destructor must not throw

The last owner will call \_\_d(\_\_p)

Definition at line 173 of file bits/shared\_ptr.h.

#### 5.1027.2.5 shared\_ptr() [5/12]

```
template<typename _Tp>
template<typename _Yp , typename _Deleter , typename _Alloc , typename = _Constructible<_Yp*, _↵
Deleter, _Alloc>>
std::shared_ptr< _Tp >::shared_ptr (
    _Yp * __p,
    _Deleter __d,
    _Alloc __a ) [inline]
```

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.

##### Parameters

<code>__↵ __p</code>	A pointer.
<code>__↵ __d</code>	A deleter.
<code>__↵ __a</code>	An allocator.

##### Postcondition

use\_count() == 1 && get() == \_\_p

##### Exceptions

<code>std::bad_alloc</code> , in	which case __d(__p) is called.
----------------------------------	--------------------------------

Requirements: \_Deleter's copy constructor and destructor must not throw \_Alloc's copy constructor and destructor must not throw.

\_\_shared\_ptr will release \_\_p by calling \_\_d(\_\_p)

Definition at line 193 of file bits/shared\_ptr.h.

**5.1027.2.6** `shared_ptr()` [6/12]

```

template<typename _Tp>
template<typename _Deleter , typename _Alloc >
std::shared_ptr< _Tp >::shared_ptr (
    nullptr_t __p,
    _Deleter __d,
    _Alloc __a ) [inline]

```

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

**Parameters**

<code>__p</code>	A null pointer constant.
<code>__d</code>	A deleter.
<code>__a</code>	An allocator.

**Postcondition**

`use_count() == 1 && get() == __p`

**Exceptions**

<code>std::bad_alloc</code> , in	which case <code>__d(__p)</code> is called.
----------------------------------	---

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

The last owner will call `__d(__p)`

Definition at line 212 of file `bits/shared_ptr.h`.

**5.1027.2.7** `shared_ptr()` [7/12]

```

template<typename _Tp>
template<typename _Yp >
std::shared_ptr< _Tp >::shared_ptr (
    const shared_ptr< _Yp > & __r,
    element_type * __p ) [inline], [noexcept]

```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

## Parameters

$\leftarrow$ __r	A shared_ptr.
$\leftarrow$ __p	A pointer that will remain valid while *__r is valid.

## Postcondition

```
get() == __p && use_count() == __r.use_count()
```

This can be used to construct a shared\_ptr to a sub-object of an object managed by an existing shared\_ptr.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 234 of file bits/shared\_ptr.h.

## 5.1027.2.8 shared\_ptr() [8/12]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<const shared_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
    const shared_ptr< _Yp > & __r ) [inline], [noexcept]
```

If \_\_r is empty, constructs an empty shared\_ptr; otherwise construct a shared\_ptr that shares ownership with \_\_r.

## Parameters

$\leftarrow$	A shared_ptr.
$\leftarrow$	
$\leftarrow$	
$\leftarrow$ r	

## Postcondition

```
get() == __r.get() && use_count() == __r.use_count()
```

Definition at line 246 of file bits/shared\_ptr.h.

## 5.1027.2.9 shared\_ptr() [9/12]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
    shared_ptr< _Tp > && __r ) [inline], [noexcept]
```

Move-constructs a shared\_ptr instance from \_\_r.



**Parameters**

↵	A shared_ptr rvalue.
↵	
↵	
↵	
<i>r</i>	

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 254 of file bits/shared\_ptr.h.

**5.1027.2.10 shared\_ptr()** [10/12]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<shared_ptr<_Yp>>>
std::shared_ptr< _Tp >::shared_ptr (
    shared_ptr< _Yp > && __r ) [inline], [noexcept]
```

Move-constructs a shared\_ptr instance from `__r`.

**Parameters**

↵	A shared_ptr rvalue.
↵	
↵	
↵	
<i>r</i>	

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 263 of file bits/shared\_ptr.h.

**5.1027.2.11 shared\_ptr()** [11/12]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<const weak_ptr<_Yp>>>
std::shared_ptr< _Tp >::shared_ptr (
    const weak_ptr< _Yp > & __r ) [inline], [explicit]
```

Constructs a shared\_ptr that shares ownership with `__r` and stores a copy of the pointer stored in `__r`.

## Parameters

↩	A weak_ptr.
↩	
↩	
↩	
<i>r</i>	

## Postcondition

```
use_count() == __r.use_count()
```

## Exceptions

<i>bad_weak_ptr</i>	when __r.expired(), in which case the constructor has no effect.
---------------------	--

Definition at line 275 of file bits/shared\_ptr.h.

## 5.1027.2.12 shared\_ptr() [12/12]

```
template<typename _Tp>
constexpr std::shared_ptr< _Tp >::shared_ptr (
    nullptr_t ) [inline], [noexcept]
```

Construct an empty shared\_ptr.

## Postcondition

```
use_count() == 0 && get() == nullptr
```

Definition at line 307 of file bits/shared\_ptr.h.

## 5.1027.3 Friends And Related Function Documentation

## 5.1027.3.1 allocate\_shared

```
template<typename _Tp>
template<typename _Yp , typename _Alloc , typename... _Args>
shared_ptr<_Yp> allocate_shared (
    const _Alloc & __a,
    _Args &&... __args ) [friend]
```

Create an object that is owned by a shared\_ptr.

**Parameters**

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

**Returns**

A `shared_ptr` that owns the newly created object.

**Exceptions**

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

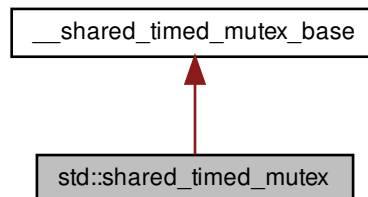
Definition at line 703 of file `bits/shared_ptr.h`.

The documentation for this class was generated from the following files:

- [bits/shared\\_ptr.h](#)
- [auto\\_ptr.h](#)

**5.1028 std::shared\_timed\_mutex Class Reference**

Inheritance diagram for `std::shared_timed_mutex`:



## Public Member Functions

- `shared_timed_mutex` (const [shared\\_timed\\_mutex](#) &)=delete
- void `lock` ()
- void `lock_shared` ()
- `shared_timed_mutex` & `operator=` (const [shared\\_timed\\_mutex](#) &)=delete
- bool `try_lock` ()
- template<typename `_Rep` , typename `_Period` >  
bool `try_lock_for` (const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rel\_time)
- bool `try_lock_shared` ()
- template<typename `_Rep` , typename `_Period` >  
bool `try_lock_shared_for` (const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rel\_time)
- template<typename `_Clock` , typename `_Duration` >  
bool `try_lock_shared_until` (const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_abs\_time)
- template<typename `_Clock` , typename `_Duration` >  
bool `try_lock_until` (const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_abs\_time)
- void `unlock` ()
- void `unlock_shared` ()

## 5.1028.1 Detailed Description

The standard shared timed mutex type.

Definition at line 359 of file `shared_mutex`.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

5.1029 `std::shuffle_order_engine<_RandomNumberEngine, __k>` Class Template Reference

## Public Types

- typedef `_RandomNumberEngine::result_type` [result\\_type](#)

## Public Member Functions

- [shuffle\\_order\\_engine](#) ()
- [shuffle\\_order\\_engine](#) (const `_RandomNumberEngine` &\_\_rng)
- [shuffle\\_order\\_engine](#) (`_RandomNumberEngine` &&\_\_rng)
- [shuffle\\_order\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` , typename = typename `std::enable_if`<!std::is\_same<`_Sseq`, [shuffle\\_order\\_engine](#)>::value && !std::is\_↔  
same<`_Sseq`, `_RandomNumberEngine`>::value> ::type>  
[shuffle\\_order\\_engine](#) (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) `operator()` ()
- void [seed](#) ()
- void [seed](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` >  
void [seed](#) (`_Sseq` &\_\_q)

### Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

### Static Public Attributes

- static constexpr size\_t **table\_size**

### Friends

- template<typename \_RandomNumberEngine1, size\_t \_\_k1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const  
[std::shuffle\\_order\\_engine](#)< \_RandomNumberEngine1, \_\_k1 > &\_\_x)
- bool [operator==](#) (const [shuffle\\_order\\_engine](#) &\_\_lhs, const [shuffle\\_order\\_engine](#) &\_\_rhs)
- template<typename \_RandomNumberEngine1, size\_t \_\_k1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is,  
[std::shuffle\\_order\\_engine](#)< \_RandomNumberEngine1, \_\_k1 > &\_\_x)

### 5.1029.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __k>  
class std::shuffle_order_engine< _RandomNumberEngine, __k >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_k.

Definition at line 1277 of file random.h.

### 5.1029.2 Member Typedef Documentation

#### 5.1029.2.1 result\_type

```
template<typename _RandomNumberEngine, size_t __k>  
typedef _RandomNumberEngine::result_type std::shuffle\_order\_engine< _RandomNumberEngine, __k >↔  
::result_type
```

The type of the generated random value.

Definition at line 1280 of file random.h.

### 5.1029.3 Constructor & Destructor Documentation

### 5.1029.3.1 shuffle\_order\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine ( ) [inline]
```

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

Definition at line 1293 of file random.h.

### 5.1029.3.2 shuffle\_order\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    const _RandomNumberEngine & __rng ) [inline], [explicit]
```

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1304 of file random.h.

### 5.1029.3.3 shuffle\_order\_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    _RandomNumberEngine && __rng ) [inline], [explicit]
```

Move constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1315 of file random.h.

**5.1029.3.4 shuffle\_order\_engine()** [4/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    result_type __s ) [inline], [explicit]
```

Seed constructs a shuffle\_order\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

**Parameters**

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 1326 of file random.h.

**5.1029.3.5 shuffle\_order\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, shuffle_order_
_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
    _Sseq & __q ) [inline], [explicit]
```

Generator construct a shuffle\_order\_engine engine.

**Parameters**

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 1340 of file random.h.

**5.1029.4 Member Function Documentation****5.1029.4.1 base()**

```
template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine& std::shuffle_order_engine< _RandomNumberEngine, __k >::base ( ) const
[inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1383 of file random.h.

#### 5.1029.4.2 `discard()`

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine<_RandomNumberEngine, __k>::discard (
    unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Definition at line 1404 of file random.h.

#### 5.1029.4.3 `max()`

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result_type std::shuffle_order_engine<_RandomNumberEngine, __k>::max ( ) [inline],
[static]
```

Gets the maximum value in the generated random number range.

Definition at line 1397 of file random.h.

References `std::max()`.

#### 5.1029.4.4 `min()`

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result_type std::shuffle_order_engine<_RandomNumberEngine, __k>::min ( ) [inline],
[static]
```

Gets the minimum value in the generated random number range.

Definition at line 1390 of file random.h.

References `std::min()`.

#### 5.1029.4.5 `operator()()`

```
template<typename _RandomNumberEngine , size_t __k>
shuffle_order_engine<_RandomNumberEngine, __k>::result_type std::shuffle_order_engine<_↵
RandomNumberEngine, __k>::operator() ( )
```

Gets the next value in the generated random number sequence.

Definition at line 815 of file bits/random.tcc.



**5.1029.4.6 seed()** [1/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ( ) [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1349 of file random.h.

**5.1029.4.7 seed()** [2/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
    result_type __s ) [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1360 of file random.h.

**5.1029.4.8 seed()** [3/3]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq >
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
    _Sseq & __q ) [inline]
```

Reseeds the shuffle\_order\_engine object with the given seed sequence.

**Parameters**

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 1373 of file random.h.

**5.1029.5 Friends And Related Function Documentation****5.1029.5.1 operator<<**

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits >
```

```
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x ) [friend]
```

Inserts the current state of a shuffle\_order\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A shuffle_order_engine random number generator engine.

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

#### 5.1029.5.2 operator==

```
template<typename _RandomNumberEngine, size_t __k>
bool operator== (
    const shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
    const shuffle_order_engine< _RandomNumberEngine, __k > & __rhs ) [friend]
```

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

#### Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1428 of file random.h.

#### 5.1029.5.3 operator>>

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x ) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

<code>__is</code>	An input stream.
<code>__X</code>	A <code>shuffle_order_engine</code> random number generator engine.

**Returns**

The input stream with the state of `__X` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.1030 std::slice Class Reference****Public Member Functions**

- [slice](#) ()
- [slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

**5.1030.1 Detailed Description**

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 59 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

**5.1031 std::slice\_array<\_Tp> Class Template Reference****Public Types**

- typedef `_Tp` **value\_type**

## Public Member Functions

- `slice_array` (const `slice_array` &)
- `void operator%=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator%=(const _Expr< _Dom, _Tp > &) const`
- `void operator&=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator&=(const _Expr< _Dom, _Tp > &) const`
- `void operator*=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator*=(const _Expr< _Dom, _Tp > &) const`
- `void operator+=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator+=(const _Expr< _Dom, _Tp > &) const`
- `void operator-=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator-=(const _Expr< _Dom, _Tp > &) const`
- `void operator/=(const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator/=(const _Expr< _Dom, _Tp > &) const`
- `void operator<=<= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `slice_array & operator=(const slice_array &)`
- `void operator= (const valarray< _Tp > &) const`
- `void operator= (const _Tp &) const`
- `template<class _Dom >`  
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `void operator>=>= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`  
`void operator|= (const _Expr< _Dom, _Tp > &) const`

## Friends

- `class valarray< _Tp >`

## 5.1031.1 Detailed Description

```
template<typename _Tp>
class std::slice_array< _Tp >
```

Reference to one-dimensional subset of an array.

A `slice_array` is a reference to the actual elements of an array specified by a slice. The way to get a `slice_array` is to call `operator[](slice)` on a `valarray`. The returned `slice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `slice_array` refers to.

## Parameters

<i>Tp</i>	Element type.
-----------	---------------

Definition at line 80 of file valarray.

The documentation for this class was generated from the following files:

- [valarray](#)
- [slice\\_array.h](#)

### 5.1032 `std::stack<_Tp, _Sequence>` Class Template Reference

#### Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

#### Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>`  
`stack ()`
- `stack (const _Sequence &__c)`
- `stack (_Sequence &&__c)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack (const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack (const _Sequence &__c, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack (_Sequence &&__c, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack (const stack &__q, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack (stack &&__q, const _Alloc &__a)`
- `template<typename... _Args>`  
`void emplace (_Args &&... __args)`
- `bool empty () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type &&__x)`
- `size_type size () const`
- `void swap (stack &__s) noexcept(__is_nothrow_swappable<_Sequence>::value)`
- `reference top ()`
- `const_reference top () const`

### Protected Attributes

- `_Sequence c`

### Friends

- `template<typename _Tp1, typename _Seq1 >`  
`bool operator< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >`  
`bool operator== (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)`

### 5.1032.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::stack< _Tp, _Sequence >
```

A standard container giving FILO behavior.

#### Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> .

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_back`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 99 of file `stl_stack.h`.

### 5.1032.2 Constructor & Destructor Documentation

#### 5.1032.2.1 stack()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↔
_Seq>::value>::type>
std::stack< _Tp, _Sequence >::stack ( ) [inline]
```

Default constructor creates no elements.

Definition at line 148 of file `stl_stack.h`.

### 5.1032.3 Member Function Documentation

#### 5.1032.3.1 empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack<_Tp, _Sequence>::empty ( ) const [inline]
```

Returns true if the stack is empty.

Definition at line 185 of file stl\_stack.h.

#### 5.1032.3.2 pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack<_Tp, _Sequence>::pop ( ) [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.


Definition at line 258 of file stl\_stack.h.

#### 5.1032.3.3 push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack<_Tp, _Sequence>::push (
    const value_type & __x ) [inline]
```

Add data to the top of the stack.

##### Parameters

 __x	Data to be added.
--	-------------------

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 225 of file stl\_stack.h.

#### 5.1032.3.4 size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack<_Tp, _Sequence>::size ( ) const [inline]
```

Returns the number of elements in the stack.

Definition at line 190 of file stl\_stack.h.

#### 5.1032.3.5 top() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack<_Tp, _Sequence>::top ( ) [inline]
```

Returns a read/write reference to the data at the first element of the stack.

Definition at line 198 of file stl\_stack.h.

#### 5.1032.3.6 top() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack<_Tp, _Sequence>::top ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 209 of file stl\_stack.h.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

## 5.1033 std::student\_t\_distribution<\_RealType> Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)



## Public Member Functions

- **student\_t\_distribution** (\_RealType \_\_n=\_RealType(1))
- **student\_t\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- \_RealType **n** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::student\\_t\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [student\\_t\\_distribution](#) &\_\_d1, const [student\\_t\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::student\\_t\\_distribution](#)< \_RealType1 > &\_\_x)

## 5.1033.1 Detailed Description

```
template<typename _RealType = double>
class std::student_t_distribution< _RealType >
```

A [student\\_t\\_distribution](#) random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3229 of file random.h.

### 5.1033.2 Member Typedef Documentation

#### 5.1033.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::student\_t\_distribution< _RealType >::result\_type
```

The type of the range of the distribution.

Definition at line 3232 of file random.h.

### 5.1033.3 Member Function Documentation

#### 5.1033.3.1 max()

```
template<typename _RealType = double>
result\_type std::student\_t\_distribution< _RealType >::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3317 of file random.h.

References [std::numeric\\_limits<\\_Tp>::max\(\)](#).

#### 5.1033.3.2 min()

```
template<typename _RealType = double>
result\_type std::student\_t\_distribution< _RealType >::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3310 of file random.h.

References [std::numeric\\_limits<\\_Tp>::lowest\(\)](#).

### 5.1033.3.3 operator()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::student_t_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 3325 of file random.h.

References `std::sqrt()`.

### 5.1033.3.4 param() [1/2]

```
template<typename _RealType = double>
param_type std::student_t_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3295 of file random.h.

### 5.1033.3.5 param() [2/2]

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3303 of file random.h.

### 5.1033.3.6 reset()

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 3278 of file random.h.

References `std::normal_distribution< _RealType >::reset()`, and `std::gamma_distribution< _RealType >::reset()`.

## 5.1033.4 Friends And Related Function Documentation

## 5.1033.4.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream<_CharT, _Traits> & __os,
    const std::student_t_distribution<_RealType1> & __x ) [friend]
```

Inserts a student\_t\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

__os	An output stream.
__x	A student_t_distribution random number distribution.

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## 5.1033.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
    const student_t_distribution<_RealType> & __d1,
    const student_t_distribution<_RealType> & __d2 ) [friend]
```

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3374 of file random.h.

## 5.1033.4.3 operator&gt;&gt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream<_CharT, _Traits> & __is,
    std::student_t_distribution<_RealType1> & __x ) [friend]
```

Extracts a student\_t\_distribution random number distribution \_\_x from the input stream \_\_is.

**Parameters**

<a href="#"><code>__is</code></a>	An input stream.
<a href="#"><code>__x</code></a>	A <code>student_t_distribution</code> random number generator engine.

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**5.1034 `std::student_t_distribution<_RealType>::param_type` Struct Reference****Public Types**

- typedef [student\\_t\\_distribution](#)<\_RealType> **distribution\_type**

**Public Member Functions**

- **param\_type** (\_RealType \_\_n=\_RealType(1))
- \_RealType **n** () const

**Friends**

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

**5.1034.1 Detailed Description**

```
template<typename _RealType = double>
struct std::student_t_distribution<_RealType>::param_type
```

Parameter type.

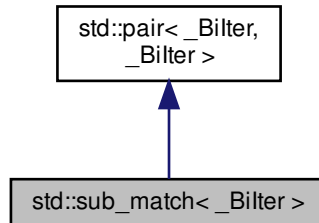
Definition at line 3239 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.1035 std::sub\_match&lt;\_Bilter&gt; Class Template Reference

Inheritance diagram for std::sub\_match<\_Bilter>:



## Public Types

- using `_PCCFP` = `_PCC<lis_same<_Bilter, _U1>::value||lis_same<_Bilter, _U2>::value, _Bilter, _Bilter>`
- using `_PCCP` = `_PCC<true, _Bilter, _Bilter>`
- typedef `__iter_traits::difference_type` **difference\_type**
- typedef `_Bilter` **first\_type**
- typedef `_Bilter` **iterator**
- typedef `_Bilter` **second\_type**
- typedef `std::basic_string<value_type>` **string\_type**
- typedef `__iter_traits::value_type` **value\_type**

## Public Member Functions

- int **compare** (const `sub_match` &\_\_s) const
- int **compare** (const `string_type` &\_\_s) const
- int **compare** (const `value_type` \*\_\_s) const
- `difference_type` **length** () const
- `operator string_type` () const
- `string_type` **str** () const
- void **swap** (`pair` &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Bilter>, \_\_is\_nothrow\_swappable<\_Bilter>>::value)

## Public Attributes

- `_Bilter` **first**
- bool **matched**
- `_Bilter` **second**

### 5.1035.1 Detailed Description

```
template<typename _BiIter>
class std::sub_match<_BiIter >
```

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with `std::basic_string` objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 864 of file `regex.h`.

### 5.1035.2 Member Typedef Documentation

#### 5.1035.2.1 \_PCCFP

```
using std::pair<_BiIter, _BiIter>::_PCCFP = _PCC<!is_same<_BiIter, _U1>::value || !is_same<↵
_BiIter, _U2>::value, _BiIter, _BiIter > [inherited]
```

There is also a templated copy ctor for the `pair` class itself.

Definition at line 283 of file `stl_pair.h`.

#### 5.1035.2.2 \_PCCP

```
using std::pair<_BiIter, _BiIter>::_PCCP = _PCC<true, _BiIter, _BiIter > [inherited]
```

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 252 of file `stl_pair.h`.

#### 5.1035.2.3 second\_type

```
typedef _BiIter std::pair<_BiIter, _BiIter>::second_type [inherited]
```

`first_type` is the first bound type

Definition at line 212 of file `stl_pair.h`.

### 5.1035.3 Member Function Documentation

#### 5.1035.3.1 compare() [1/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter>::compare (
    const sub_match<_BiIter > & __s ) const [inline]
```

Compares this and another matched sequence.

## Parameters

<code>__s</code>	Another matched sequence to compare to this one.
------------------	--

## Return values

<code>&lt;0</code>	this matched sequence will collate before <code>__s</code> .
<code>=0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>&gt;0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 925 of file regex.h.

Referenced by `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

## 5.1035.3.2 compare() [2/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter >::compare (
    const string_type & __s ) const [inline]
```

Compares this `sub_match` to a string.

## Parameters

<code>__s</code>	A string to compare to this <code>sub_match</code> .
------------------	--

## Return values

<code>&lt;0</code>	this matched sequence will collate before <code>__s</code> .
<code>=0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>&gt;0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 938 of file regex.h.

## 5.1035.3.3 compare() [3/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter >::compare (
    const value_type * __s ) const [inline]
```

Compares this `sub_match` to a C-style string.



**Parameters**

<code>__s</code>	A C-style string to compare to this <code>sub_match</code> .
------------------	--

**Return values**

<code>&lt; 0</code>	this matched sequence will collate before <code>__s</code> .
<code>= 0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>&gt; 0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 951 of file `regex.h`.

**5.1035.3.4 `length()`**

```
template<typename _BiIter>
difference_type std::sub_match< _BiIter >::length ( ) const [inline]
```

Gets the length of the matching sequence.

Definition at line 882 of file `regex.h`.

**5.1035.3.5 `operator string_type()`**

```
template<typename _BiIter>
std::sub_match< _BiIter >::operator string_type ( ) const [inline]
```

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the `str()` member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 895 of file `regex.h`.

### 5.1035.3.6 str()

```
template<typename _BiIter>
string_type std::sub_match< _BiIter >::str ( ) const [inline]
```

Gets the matching sequence as a string.

#### Returns

the matching sequence as a string.

Definition at line 908 of file regex.h.

Referenced by std::sub\_match< \_Bi\_iter >::compare(), and std::operator<<().

### 5.1035.4 Member Data Documentation

#### 5.1035.4.1 first

```
_BiIter std::pair< _BiIter , _BiIter >::first [inherited]
```

second\_type is the second bound type

Definition at line 214 of file stl\_pair.h.

#### 5.1035.4.2 second

```
_BiIter std::pair< _BiIter , _BiIter >::second [inherited]
```

first is a copy of the first object

Definition at line 215 of file stl\_pair.h.

The documentation for this class was generated from the following file:

- [regex.h](#)

### 5.1036 std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r > Class Template Reference

#### Public Types

- typedef \_UIntType [result\\_type](#)

## Public Member Functions

- [subtract\\_with\\_carry\\_engine](#) ([result\\_type](#) \_\_sd=default\_seed)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, subtract_with_carry_engine>::value>::type>`  
[subtract\\_with\\_carry\\_engine](#) (\_Sseq &\_\_q)
- `void` [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- `void` [seed](#) ([result\\_type](#) \_\_sd=default\_seed)
- `template<typename _Sseq >`  
`std::enable_if< std::is_class<_Sseq>::value >::type` [seed](#) (\_Sseq &\_\_q)

## Static Public Member Functions

- `static constexpr` [result\\_type](#) [max](#) ()
- `static constexpr` [result\\_type](#) [min](#) ()

## Static Public Attributes

- `static constexpr` [result\\_type](#) [default\\_seed](#)
- `static constexpr` `size_t` [long\\_lag](#)
- `static constexpr` `size_t` [short\\_lag](#)
- `static constexpr` `size_t` [word\\_size](#)

## Friends

- `template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits> &` [operator<<](#) (`std::basic_ostream<_CharT, _Traits> &__os`, `const`  
`std::subtract_with_carry_engine<_UIntType1, __w1, __s1, __r1> &__x`)
- `bool` [operator==](#) (`const` [subtract\\_with\\_carry\\_engine](#) &\_\_lhs, `const` [subtract\\_with\\_carry\\_engine](#) &\_\_rhs)
- `template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits >`  
`std::basic_istream<_CharT, _Traits> &` [operator>>](#) (`std::basic_istream<_CharT, _Traits> &__is`,  
`std::subtract_with_carry_engine<_UIntType1, __w1, __s1, __r1> &__x`)

### 5.1036.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
class std::subtract_with_carry_engine<_UIntType, __w, __s, __r>
```

The Marsaglia-Zaman generator.

This is a model of a Generalized Fibonacci discrete random number generator, sometimes referred to as the SWC generator.

A discrete random number generator that produces pseudorandom numbers using:

$$x_i \leftarrow (x_{i-s} - x_{i-r} - carry_{i-1}) \bmod m$$

The size of the state is  $r$  and the maximum period of the generator is  $(m^r - m^s - 1)$ .

Definition at line 652 of file random.h.

## 5.1036.2 Member Typedef Documentation

## 5.1036.2.1 result\_type

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
typedef _UIntType std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type
```

The type of the generated random value.

Definition at line 655 of file random.h.

## 5.1036.3 Constructor &amp; Destructor Documentation

## 5.1036.3.1 subtract\_with\_carry\_engine() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
    result_type __sd = default_seed ) [inline], [explicit]
```

Constructs an explicitly seeded % subtract\_with\_carry\_engine random number generator.

Definition at line 676 of file random.h.

References std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >::seed().

## 5.1036.3.2 subtract\_with\_carry\_engine() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, subtract_with_carry_engine>::value> ::type>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
    _Sseq & __q ) [inline], [explicit]
```

Constructs a subtract\_with\_carry\_engine random number engine seeded from the seed sequence \_\_q.

## Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

Definition at line 689 of file random.h.

References `std::subtract_with_carry_engine<_UIntType, __w, __s, __r>::seed()`.

#### 5.1036.4 Member Function Documentation

##### 5.1036.4.1 `discard()`

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine<_UIntType, __w, __s, __r>::discard (
    unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Definition at line 735 of file `random.h`.

##### 5.1036.4.2 `max()`

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine<_UIntType, __w, __s, __r>::max ( )
[inline], [static]
```

Gets the inclusive maximum value of the range of random integers returned by this generator.

Definition at line 728 of file `random.h`.

##### 5.1036.4.3 `min()`

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine<_UIntType, __w, __s, __r>::min ( )
[inline], [static]
```

Gets the inclusive minimum value of the range of random integers returned by this generator.

Definition at line 720 of file `random.h`.

##### 5.1036.4.4 `operator()()`

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
subtract_with_carry_engine<_UIntType, __w, __s, __r>::result_type std::subtract_with_carry_engine<
_UIntType, __w, __s, __r>::operator() ( )
```

Gets the next random number in the sequence.

Definition at line 595 of file `bits/random.tcc`.

**5.1036.4.5 seed()** [1/2]

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
    result_type __sd = default_seed )
```

Seeds the initial state  $x_0$  of the random number generator.

N1688[4.19] modifies this as follows. If `__value == 0`, sets value to 19780503. In any case, with a linear congruential generator  $lcg(i)$  having parameters  $m_{lcg} = 2147483563$ ,  $a_{lcg} = 40014$ ,  $c_{lcg} = 0$ , and  $lcg(0) = value$ , sets  $x_{-r} \dots x_{-1}$  to  $lcg(1) \bmod m \dots lcg(r) \bmod m$  respectively. If  $x_{-1} = 0$  set carry to 1, otherwise sets carry to 0.

Definition at line 540 of file bits/random.tcc.

Referenced by `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine()`.

**5.1036.4.6 seed()** [2/2]

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
template<typename _Sseq >
std::enable_if< std::is_class< _Sseq >::value >::type std::subtract_with_carry_engine< _UIntType,
__w, __s, __r >::seed (
    _Sseq & __q )
```

Seeds the initial state  $x_0$  of the % `subtract_with_carry_engine` random number generator.

Definition at line 569 of file bits/random.tcc.

**5.1036.5 Friends And Related Function Documentation****5.1036.5.1 operator<<**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1 , size_t __w1, size_t __s1, size_t __r1, typename _CharT , typename
_Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x ) [friend]
```

Inserts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` into the output stream `__os`.

**Parameters**

<code>__os</code>	An output stream.
<code>__x</code>	A % <code>subtract_with_carry_engine</code> random number generator engine.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**5.1036.5.2 operator==**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool operator== (
    const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
    const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs ) [friend]
```

Compares two % subtract\_with\_carry\_engine random number generator objects of the same type for equality.

**Parameters**

<code>__lhs</code>	A % subtract_with_carry_engine random number generator object.
<code>__rhs</code>	Another % subtract_with_carry_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 760 of file random.h.

**5.1036.5.3 operator>>**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename
_traits >
std::basic_istream<_CharT, _Traits>& operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x ) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

**Parameters**

<code>__is</code>	An input stream.
<code>__x</code>	A % subtract_with_carry_engine random number generator engine.

## Returns

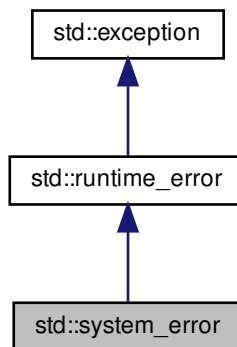
The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.1037 std::system\_error Class Reference

Inheritance diagram for `std::system_error`:



## Public Member Functions

- **system\_error** ([error\\_code](#) \_\_ec=[error\\_code](#)())
- **system\_error** ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- **system\_error** ([error\\_code](#) \_\_ec, const char \*\_\_what)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat, const char \*\_\_what)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat, const [string](#) &\_\_what)
- const [error\\_code](#) & **code** () const noexcept
- virtual const char \* **what** () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.1037.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 341 of file `system_error`.



## 5.1037.2 Member Function Documentation

### 5.1037.2.1 what()

```
virtual const char* std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.1038 std::thread Class Reference

### Classes

- class [id](#)

### Public Types

- using **\_State\_ptr** = [unique\\_ptr](#)< \_State >
- typedef \_\_gthread\_t **native\_handle\_type**

### Public Member Functions

- template<typename \_Callable , typename... \_Args, typename = \_Require<\_\_not\_same<\_Callable>>>>  
**thread** (\_Callable &&\_\_f, \_Args &&... \_\_args)
- **thread** (const [thread](#) &)=delete
- **thread** ([thread](#) &&\_\_t) noexcept
- void **detach** ()
- [thread::id](#) **get\_id** () const noexcept
- void **join** ()
- bool **joinable** () const noexcept
- native\_handle\_type **native\_handle** ()
- [thread](#) & **operator=** (const [thread](#) &)=delete
- [thread](#) & **operator=** ([thread](#) &&\_\_t) noexcept
- void **swap** ([thread](#) &&\_\_t) noexcept

### Static Public Member Functions

- template<typename \_Callable , typename... \_Args>  
static \_Invoker< [\\_\\_decayed\\_tuple](#)< \_Callable, \_Args... > > **\_\_make\_invoker** (\_Callable &&\_\_callable, \_Args &&... \_\_args)
- static unsigned int **hardware\_concurrency** () noexcept

### 5.1038.1 Detailed Description

thread

Definition at line 62 of file thread.

### 5.1038.2 Member Function Documentation

#### 5.1038.2.1 native\_handle()

```
native_handle_type std::thread::native_handle ( ) [inline]
```

#### Precondition

thread is joinable

Definition at line 179 of file thread.

The documentation for this class was generated from the following file:

- [thread](#)

## 5.1039 std::thread::id Class Reference

### Public Member Functions

- **id** (native\_handle\_type \_\_id)

### Friends

- class **hash**< **thread::id** >
- bool **operator**< ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- template<class \_CharT, class \_Traits >  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator**<< ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_out, [thread::id](#) \_\_id)
- bool **operator**== ([thread::id](#) \_\_x, [thread::id](#) \_\_y) noexcept
- class **thread**

### 5.1039.1 Detailed Description

thread::id

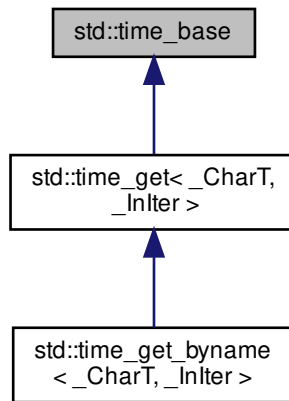
Definition at line 77 of file thread.

The documentation for this class was generated from the following file:

- [thread](#)

## 5.1040 std::time\_base Class Reference

Inheritance diagram for std::time\_base:



### Public Types

- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }

### 5.1040.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

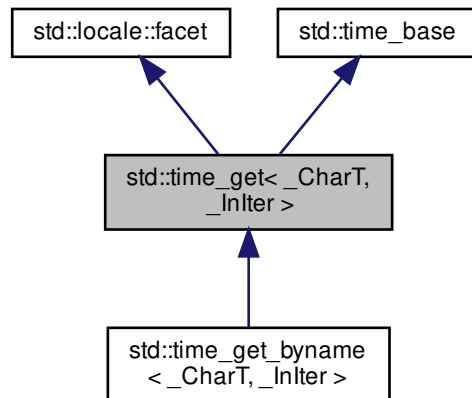
Definition at line 52 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.1041 std::time\_get&lt; \_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for std::time\_get< \_CharT, \_InIter >:



## Public Types

- enum **dateorder** {  
  **no\_order**, **dmy**, **mdy**, **ymd**,  
  **ydm** }
- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

## Public Member Functions

- `time_get` (`size_t __refs=0`)
- `dateorder date_order` () const
- `iter_type get` (`iter_type __s`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`, `char __format`, `char __modifier=0`) const
- `iter_type get` (`iter_type __s`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`, const `char_type * __fmt`, const `char_type * __fmtend`) const
- `iter_type get_date` (`iter_type __beg`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`) const
- `iter_type get_monthname` (`iter_type __beg`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`) const
- `iter_type get_time` (`iter_type __beg`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`) const
- `iter_type get_weekday` (`iter_type __beg`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`) const
- `iter_type get_year` (`iter_type __beg`, `iter_type __end`, `ios_base & __io`, `ios_base::iostate & __err`, `tm * __tm`) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~time\\_get](#) ()
- [iter\\_type\\_M\\_extract\\_name](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type\\_M\\_extract\\_num](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type\\_M\\_extract\\_via\\_format](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format) const
- [iter\\_type\\_M\\_extract\\_wday\\_or\\_month](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- virtual dateorder [do\\_date\\_order](#) () const
- [iter\\_type do\\_get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_f, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, char \*\_\_format, char \_\_modifier) const
- virtual [iter\\_type do\\_get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 5.1041.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get< _CharT, _InIter >
```

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The `time_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_get` facet.

Definition at line 368 of file `locale_facets_nonio.h`.

## 5.1041.2 Member Typedef Documentation

### 5.1041.2.1 char\_type

```
template<typename _CharT , typename _InIter >
typedef _CharT std::time_get< _CharT, _InIter >::char_type
```

Public typedefs.

Definition at line 374 of file locale\_facets\_nonio.h.

### 5.1041.2.2 iter\_type

```
template<typename _CharT , typename _InIter >
typedef _InIter std::time_get< _CharT, _InIter >::iter_type
```

Public typedefs.

Definition at line 375 of file locale\_facets\_nonio.h.

## 5.1041.3 Constructor & Destructor Documentation

### 5.1041.3.1 time\_get()

```
template<typename _CharT , typename _InIter >
std::time_get< _CharT, _InIter >::time_get (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 389 of file locale\_facets\_nonio.h.

### 5.1041.3.2 ~time\_get()

```
template<typename _CharT , typename _InIter >
virtual std::time_get< _CharT, _InIter >::~~time_get ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 593 of file locale\_facets\_nonio.h.

## 5.1041.4 Member Function Documentation

### 5.1041.4.1 date\_order()

```
template<typename _CharT , typename _InIter >
dateorder std::time_get< _CharT, _InIter >::date_order ( ) const [inline]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_↵put::put() only uses month, day, and year. If the format x for the associated locale uses other fields, this function returns time\_base::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

#### Returns

A member of time\_base::dateorder.

Definition at line 406 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_date\_order().

### 5.1041.4.2 do\_date\_order()

```
template<typename _CharT , typename _InIter >
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↵
date_order ( ) const [protected], [virtual]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_↵put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of time\_base::dateorder.

Definition at line 627 of file locale\_facets\_nonio.tcc.

Referenced by std::time\_get< \_CharT, \_InIter >::date\_order().

## 5.1041.4.3 do\_get()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    ios_base & __f,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier ) const [inline], [protected]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

## See also

get() for more details.

## Parameters

<code>__s</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__f</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.
<code>__format</code>	Format specifier.
<code>__modifier</code>	Format modifier.

## Returns

Iterator to first char not parsed.

Definition at line 1241 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.



#### 5.1041.4.4 do\_get\_date()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [protected], [virtual]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

`get_date()` for details.

##### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

##### Returns

Iterator to first char beyond date string.

Definition at line 1077 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_date()`.

#### 5.1041.4.5 do\_get\_monthname()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [protected], [virtual]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

get\_monthname() for details.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond month name.

Definition at line 1120 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_monthname()`.

#### 5.1041.4.6 do\_get\_time()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [protected], [virtual]
```

Parse input time string.

This function parses a time according to the format `x` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

get\_time() for details.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond time string.

Definition at line 1060 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_time()`.

**5.1041.4.7 do\_get\_weekday()**

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const    [protected], [virtual]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_weekday()` for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond weekday name.

Definition at line 1094 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_weekday()`.

## 5.1041.4.8 do\_get\_year()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [protected], [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_year() for details.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond year.

Definition at line 1146 of file locale\_facets\_nonio.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_year()`.

## 5.1041.4.9 get() [1/2]

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier = 0 ) const [inline]
```

Parse input string according to format.

This function calls `time_get::do_get` with the provided parameters.

**See also**

`do_get()` and `get()`.

**Parameters**

<code>__s</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.
<code>__format</code>	Format specifier.
<code>__modifier</code>	Format modifier.

**Returns**

Iterator to first char not parsed.

Definition at line 559 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get()`.

**5.1041.4.10 `get()` [2/2]**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    const char_type * __fmt,
    const char_type * __fmtend ) const [inline]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of `time_put::put`. The format string follows the format specified for `strptime(3)/strptime(3)`. The actual parsing is done by `time_get::do_get`.

**Parameters**

<code>__s</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.
<code>__fmt</code>	Start of the format string.
<code>__fmtend</code>	End of the format string.

**Returns**

Iterator to first char not parsed.

Definition at line 1169 of file locale\_facets\_nonio.tcc.

References std::ios\_base::M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::\_\_ctype\_abstract\_base< \_CharT >::is(), std::\_\_ctype\_abstract\_base< \_CharT >::narrow(), std::\_\_ctype\_abstract\_base< \_CharT >::tolower(), and std::\_\_ctype\_abstract\_base< \_CharT >::toupper().

**5.1041.4.11 get\_date()**

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling time\_get::do\_get\_date().

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, *err* |= ios\_base::failbit. If parsing reads all the characters, *err* |= ios\_base::eofbit.

**Parameters**

<i>__beg</i>	Start of string to parse.
<i>__end</i>	End of string to parse.
<i>__io</i>	Source of the locale.
<i>__err</i>	Error flags to set.
<i>__tm</i>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond date string.

Definition at line 455 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_date().

#### 5.1041.4.12 `get_monthname()`

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

##### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

##### Returns

Iterator to first char beyond month name.

Definition at line 512 of file `locale_facets_nonio.h`.

References `std::time_get< _CharT, _InIter >::do_get_monthname()`.

#### 5.1041.4.13 `get_time()`

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

#### Returns

Iterator to first char beyond time string.

Definition at line 430 of file `locale_facets_nonio.h`.

References `std::time_get< _CharT, _InIter >::do_get_time()`.

#### 5.1041.4.14 get\_weekday()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.



**Returns**

Iterator to first char beyond weekday name.

Definition at line 483 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter >::do_get_weekday()`.

**5.1041.4.15 get\_year()**

```
template<typename _CharT, typename _InIter >
iter_type std::time_get<_CharT, _InIter >::get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond year.

Definition at line 538 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter >::do_get_year()`.

**5.1041.5 Member Data Documentation**

## 5.1041.5.1 id

```
template<typename _CharT , typename _InIter >
locale::id std::time_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

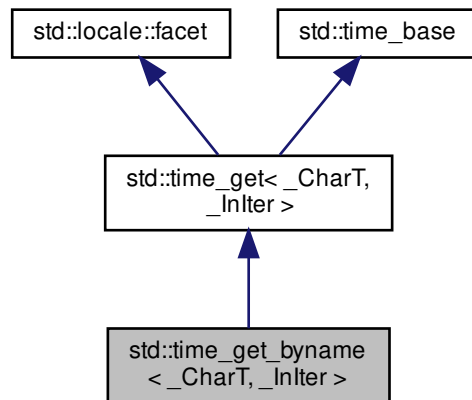
Definition at line 379 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.1042 std::time\_get\_byname&lt;\_CharT, \_InIter&gt; Class Template Reference

Inheritance diagram for std::time\_get\_byname<\_CharT, \_InIter>:



## Public Types

- typedef `_CharT` **char\_type**
- enum **dateorder** {  
  **no\_order**, **dmy**, **mdy**, **ymd**,  
  **ydm** }
- typedef `_InIter` **iter\_type**

## Public Member Functions

- **time\_get\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_get\_byname** (const string & \_\_s, size\_t \_\_refs=0)
- dateorder **date\_order** () const
- **iter\_type get** (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, char \_\_format, char \_\_modifier=0) const
- **iter\_type get** (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const char\_type \* \_\_fmt, const char\_type \* \_\_fmtend) const
- **iter\_type get\_date** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- **iter\_type get\_monthname** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- **iter\_type get\_time** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- **iter\_type get\_weekday** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- **iter\_type get\_year** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const

## Static Public Attributes

- static locale::id id

## Protected Member Functions

- **iter\_type \_M\_extract\_name** (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, const \_CharT \*\* \_\_names, size\_t \_\_indexlen, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- **iter\_type \_M\_extract\_num** (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, int \_\_min, int \_\_max, size\_t \_\_len, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- **iter\_type \_M\_extract\_via\_format** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const \_CharT \* \_\_format) const
- **iter\_type \_M\_extract\_wday\_or\_month** (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, const \_CharT \*\* \_\_names, size\_t \_\_indexlen, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- virtual dateorder **do\_date\_order** () const
- **iter\_type do\_get** (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_f, ios\_base::iostate & \_\_err, tm \* \_\_tm, char \_\_format, char \_\_modifier) const
- virtual **iter\_type do\_get\_date** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual **iter\_type do\_get\_monthname** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual **iter\_type do\_get\_time** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual **iter\_type do\_get\_weekday** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual **iter\_type do\_get\_year** (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## 5.1042.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get_byname< _CharT, _InIter >
```

class time\_get\_byname [22.2.5.2].

Definition at line 760 of file locale\_facets\_nonio.h.

## 5.1042.2 Member Function Documentation

## 5.1042.2.1 date\_order()

```
template<typename _CharT , typename _InIter >
dateorder std::time_get< _CharT, _InIter >::date_order ( ) const [inline], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_↵put::put() only uses month, day, and year. If the format x for the associated locale uses other fields, this function returns time\_base::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

## Returns

A member of time\_base::dateorder.

Definition at line 406 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_date\_order().

## 5.1042.2.2 do\_date\_order()

```
template<typename _CharT , typename _InIter >
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↵
date_order ( ) const [protected], [virtual], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_↵put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

## Returns

A member of time\_base::dateorder.

Definition at line 627 of file locale\_facets\_nonio.tcc.

Referenced by std::time\_get< \_CharT, \_InIter >::date\_order().

### 5.1042.2.3 do\_get()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get (
    iter_type __s,
    iter_type __end,
    ios_base & __f,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier ) const    [inline], [protected], [inherited]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

`get()` for more details.

#### Parameters

<code>__s</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__f</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.
<code>__format</code>	Format specifier.
<code>__modifier</code>	Format modifier.

#### Returns

Iterator to first char not parsed.

Definition at line 1241 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, and `std::__ctype_abstract_base<_CharT >::widen()`.

Referenced by `std::time_get<_CharT, _InIter >::get()`.

## 5.1042.2.4 do\_get\_date()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const    [protected], [virtual], [inherited]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_date() for details.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond date string.

Definition at line 1077 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get<\_CharT, \_InIter>::get\_date().

## 5.1042.2.5 do\_get\_monthname()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const    [protected], [virtual], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_monthname()` for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1120 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_monthname()`.

#### 5.1042.2.6 `do_get_time()`

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const    [protected], [virtual], [inherited]
```

Parse input time string.

This function parses a time according to the format `x` and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_time()` for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond time string.

Definition at line 1060 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_time().

**5.1042.2.7 do\_get\_weekday()**

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const    [protected], [virtual], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

get\_weekday() for details.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond weekday name.

Definition at line 1094 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_weekday().



### 5.1042.2.8 do\_get\_year()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const    [protected], [virtual], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

`get_year()` for details.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond year.

Definition at line 1146 of file `locale_facets_nonio.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_year()`.

### 5.1042.2.9 get() [1/2]

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    char __format,
    char __modifier = 0 ) const    [inline], [inherited]
```

Parse input string according to format.

This function calls `time_get::do_get` with the provided parameters.

See also

do\_get() and get().

#### Parameters

<code>__s</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.
<code>__format</code>	Format specifier.
<code>__modifier</code>	Format modifier.

#### Returns

Iterator to first char not parsed.

Definition at line 559 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get().

#### 5.1042.2.10 get() [2/2]

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::get (
    iter_type __s,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm,
    const char_type * __fmt,
    const char_type * __fmtend ) const [inline], [inherited]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of time\_put::put. The format string follows the format specified for strftime(3)/strptime(3). The actual parsing is done by time\_get::do\_get.

#### Parameters

<code>__s</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.
<code>__fmt</code>	Start of the format string.
<code>__fmtend</code>	End of the format string.

**Returns**

Iterator to first char not parsed.

Definition at line 1169 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::__ctype_abstract_base<_CharT>::is()`, `std::__ctype_abstract_base<_CharT>::narrow()`, `std::__ctype_abstract_base<_CharT>::tolower()`, and `std::__ctype_abstract_base<_CharT>::toupper()`.

**5.1042.2.11 get\_date()**

```
template<typename _CharT, typename _InIter >
iter_type std::time_get<_CharT, _InIter>::get_date (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline], [inherited]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

**Returns**

Iterator to first char beyond date string.

Definition at line 455 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter>::do_get_date()`.

## 5.1042.2.12 get\_monthname()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_monthname (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_monthname().

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

## Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

## Returns

Iterator to first char beyond month name.

Definition at line 512 of file locale\_facets\_nonio.h.

References std::time\_get<\_CharT, \_InIter>::do\_get\_monthname().

## 5.1042.2.13 get\_time()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_time (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline], [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

#### Returns

Iterator to first char beyond time string.

Definition at line 430 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get_time()`.

#### 5.1042.2.14 `get_weekday()`

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_weekday (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

**Returns**

Iterator to first char beyond weekday name.

Definition at line 483 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_weekday().

**5.1042.2.15 get\_year()**

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_year (
    iter_type __beg,
    iter_type __end,
    ios_base & __io,
    ios_base::iostate & __err,
    tm * __tm ) const [inline], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

**Parameters**

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond year.

Definition at line 538 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_year().

**5.1042.3 Member Data Documentation**

### 5.1042.3.1 id

```
template<typename _CharT , typename _InIter >
locale::id std::time_get< _CharT, _InIter >::id [static], [inherited]
```

Numpunct facet id.

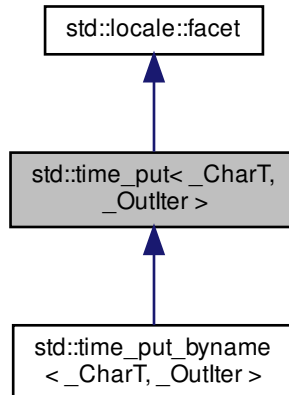
Definition at line 379 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.1043 std::time\_put< \_CharT, \_OutIter > Class Template Reference

Inheritance diagram for std::time\_put< \_CharT, \_OutIter >:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)

## Public Member Functions

- `time_put` (size\_t \_\_refs=0)
- `iter_type put` (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, const \_CharT \*\_\_beg, const \_CharT \*\_\_end) const
- `iter_type put` (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const

## Static Public Attributes

- static `locale::id` id

## Protected Member Functions

- virtual `~time_put` ()
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_c\_locale &\_\_cloc, const char \*\_\_s, \_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_c\_locale &\_\_cloc)
- static `_c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `_c_locale _S_lc_ctype_c_locale` (\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.1043.1 Detailed Description

```
template<typename _CharT, typename _Outiter>
class std::time_put< _CharT, _Outiter >
```

Primary class template `time_put`.

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

The `time_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_put` facet.

Definition at line 797 of file `locale_facets_nonio.h`.

## 5.1043.2 Member Typedef Documentation



### 5.1043.2.1 char\_type

```
template<typename _CharT , typename _OutIter >
typedef _CharT std::time_put< _CharT, _OutIter >::char_type
```

Public typedefs.

Definition at line 803 of file locale\_facets\_nonio.h.

### 5.1043.2.2 iter\_type

```
template<typename _CharT , typename _OutIter >
typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

Definition at line 804 of file locale\_facets\_nonio.h.

## 5.1043.3 Constructor & Destructor Documentation

### 5.1043.3.1 time\_put()

```
template<typename _CharT , typename _OutIter >
std::time_put< _CharT, _OutIter >::time_put (
    size_t __refs = 0 ) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 818 of file locale\_facets\_nonio.h.

### 5.1043.3.2 ~time\_put()

```
template<typename _CharT , typename _OutIter >
virtual std::time_put< _CharT, _OutIter >::~time_put ( ) [inline], [protected], [virtual]
```

Destructor.

Definition at line 864 of file locale\_facets\_nonio.h.

## 5.1043.4 Member Function Documentation

## 5.1043.4.1 do\_put()

```
template<typename _CharT, typename _OutIter >
_OutIter std::time_put< _CharT, _OutIter >::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod ) const [protected], [virtual]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

## See also

put() for more details.

## Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

## Returns

Iterator after writing.

Definition at line 1309 of file locale\_facets\_nonio.tcc.

References std::ios\_base::M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

Referenced by std::time\_put< \_CharT, \_OutIter >::put().

**5.1043.4.2** `put()` [1/2]

```
template<typename _CharT , typename _OutIter >
_OutIter std::time_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    const _CharT * __beg,
    const _CharT * __end ) const
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

**Parameters**

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__beg</code>	Start of format string.
<code>__end</code>	End of format string.

**Returns**

Iterator after writing.

Definition at line 1274 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base< _CharT >::narrow()`.

**5.1043.4.3** `put()` [2/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::time_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod = 0 ) const [inline]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

## Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct <code>tm</code> with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

## Returns

Iterator after writing.

Definition at line 857 of file `locale_facets_nonio.h`.

References `std::time_put< _CharT, _OutIter >::do_put()`.

## 5.1043.5 Member Data Documentation

## 5.1043.5.1 id

```
template<typename _CharT , typename _OutIter >
locale::id std::time_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

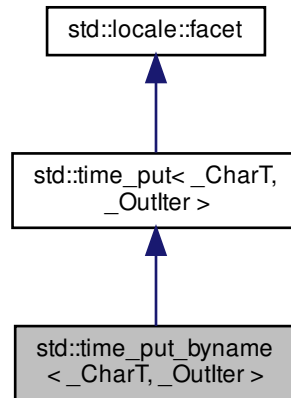
Definition at line 808 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.1044 std::time\_put\_byname<\_CharT, \_Outlter > Class Template Reference

Inheritance diagram for std::time\_put\_byname<\_CharT, \_Outlter >:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Outlter` **iter\_type**

### Public Member Functions

- **time\_put\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_put\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- [iter\\_type](#) **put** ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, const `_CharT` \*\_\_beg, const `_CharT` \*\_\_end) const
- [iter\\_type](#) **put** ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const

### Static Public Attributes

- static [locale::id](#) **id**

### Protected Member Functions

- virtual [iter\\_type](#) **do\_put** ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 5.1044.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::time_put_byname<_CharT, _OutIter>
```

class time\_put\_byname [22.2.5.4].

Definition at line 893 of file locale\_facets\_nonio.h.

## 5.1044.2 Member Function Documentation

## 5.1044.2.1 do\_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put<_CharT, _OutIter>::do_put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod) const [protected], [virtual], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

## See also

put() for more details.

## Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

**Returns**

Iterator after writing.

Definition at line 1309 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base<_CharT>::widen()`.

Referenced by `std::time_put<_CharT, _OutIter>::put()`.

**5.1044.2.2 put()** [1/2]

```
template<typename _CharT , typename _OutIter >
_OutIter std::time_put<_CharT, _OutIter>::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    const _CharT * __beg,
    const _CharT * __end ) const [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

**Parameters**

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__beg</code>	Start of format string.
<code>__end</code>	End of format string.

**Returns**

Iterator after writing.

Definition at line 1274 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base<_CharT>::narrow()`.

## 5.1044.2.3 put() [2/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::time_put< _CharT, _OutIter >::put (
    iter_type __s,
    ios_base & __io,
    char_type __fill,
    const tm * __tm,
    char __format,
    char __mod = 0 ) const [inline], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

## Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

## Returns

Iterator after writing.

Definition at line 857 of file locale\_facets\_nonio.h.

References std::time\_put<\_CharT, \_OutIter >::do\_put().

## 5.1044.3 Member Data Documentation

## 5.1044.3.1 id

```
template<typename _CharT , typename _OutIter >
locale::id std::time_put< _CharT, _OutIter >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 808 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)



## 5.1045 std::timed\_mutex Class Reference

### Public Member Functions

- **timed\_mutex** (const [timed\\_mutex](#) &)=delete
- void **lock** ()
- [timed\\_mutex](#) & **operator=** (const [timed\\_mutex](#) &)=delete
- bool **try\_lock** ()
- template<typename \_Rep , typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock , typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### 5.1045.1 Detailed Description

timed\_mutex

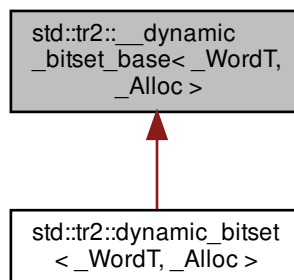
Definition at line 298 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.1046 std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc > Struct Template Reference

Inheritance diagram for std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc >:



### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_WordT **block\_type**
- typedef size\_t **size\_type**

## Public Member Functions

- `__dynamic_bitset_base` (const allocator\_type &\_\_alloc)
- `__dynamic_bitset_base` (const `__dynamic_bitset_base` &)=default
- `__dynamic_bitset_base` (`__dynamic_bitset_base` &&\_\_b)=default
- `__dynamic_bitset_base` (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type())
- `size_t _M_are_all_aux` () const noexcept
- `void _M_clear` () noexcept
- `void _M_do_and` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `void _M_do_append_block` (block\_type \_\_block, size\_type \_\_pos)
- `size_t _M_do_count` () const noexcept
- `void _M_do_dif` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `size_type _M_do_find_first` (size\_t \_\_not\_found) const
- `size_type _M_do_find_next` (size\_t \_\_prev, size\_t \_\_not\_found) const
- `void _M_do_flip` () noexcept
- `void _M_do_left_shift` (size\_t \_\_shift)
- `void _M_do_or` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `void _M_do_reset` () noexcept
- `void _M_do_right_shift` (size\_t \_\_shift)
- `void _M_do_set` () noexcept
- `unsigned long long _M_do_to_ullong` () const
- `unsigned long _M_do_to_ulong` () const
- `void _M_do_xor` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `allocator_type _M_get_allocator` () const noexcept
- `block_type & _M_getword` (size\_type \_\_pos) noexcept
- `block_type _M_getword` (size\_type \_\_pos) const noexcept
- `block_type & _M_hiword` () noexcept
- `block_type _M_hiword` () const noexcept
- `bool _M_is_any` () const noexcept
- `bool _M_is_equal` (const `__dynamic_bitset_base` &\_\_x) const noexcept
- `bool _M_is_less` (const `__dynamic_bitset_base` &\_\_x) const noexcept
- `bool _M_is_proper_subset_of` (const `__dynamic_bitset_base` &\_\_b) const noexcept
- `bool _M_is_subset_of` (const `__dynamic_bitset_base` &\_\_b) const noexcept
- `void _M_resize` (size\_t \_\_nbits, bool \_\_value)
- `size_type _M_size` () const noexcept
- `void _M_swap` (`__dynamic_bitset_base` &\_\_b) noexcept
- `__dynamic_bitset_base & operator=` (const `__dynamic_bitset_base` &)=default
- `__dynamic_bitset_base & operator=` (`__dynamic_bitset_base` &&)=default

## Static Public Member Functions

- `static block_type _S_maskbit` (size\_type \_\_pos) noexcept
- `static size_type _S_whichbit` (size\_type \_\_pos) noexcept
- `static size_type _S_whichbyte` (size\_type \_\_pos) noexcept
- `static size_type _S_whichword` (size\_type \_\_pos) noexcept

## Public Attributes

- `std::vector< block_type, allocator_type > _M_w`

### Static Public Attributes

- static const size\_type **\_S\_bits\_per\_block**
- static const size\_type **npos**

#### 5.1046.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
```

Base class, general case.

See documentation for `dynamic_bitset`.

Definition at line 62 of file `dynamic_bitset`.

#### 5.1046.2 Member Data Documentation

##### 5.1046.2.1 `_M_w`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w
```

0 is the least significant word.

Definition at line 75 of file `dynamic_bitset`.

The documentation for this struct was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

### 5.1047 `std::tr2::__reflection_typelist< _Elements >` Struct Template Reference

#### 5.1047.1 Detailed Description

```
template<typename... _Elements>
struct std::tr2::__reflection_typelist< _Elements >
```

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.

Definition at line 56 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.1048 `std::tr2::__reflection_typelist< _First, _Rest... >` Struct Template Reference

### Public Types

- typedef [std::false\\_type](#) **empty**

#### 5.1048.1 Detailed Description

```
template<typename _First, typename... _Rest>
struct std::tr2::__reflection_typelist< _First, _Rest... >
```

Partial specialization.

Definition at line 67 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.1049 `std::tr2::__reflection_typelist<>` Struct Template Reference

### Public Types

- typedef [std::true\\_type](#) **empty**

#### 5.1049.1 Detailed Description

```
template<>
struct std::tr2::__reflection_typelist<>
```

Specialization for an empty typelist.

Definition at line 60 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.1050 `std::tr2::bases< _Tp >` Struct Template Reference

### Public Types

- typedef [\\_\\_reflection\\_typelist< \\_\\_bases\(\\_Tp\)... >](#) **type**

### 5.1050.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::bases< _Tp >
```

Sequence abstraction metafunctions for manipulating a typelist.

Enumerate all the base classes of a class. Form of a typelist.

Definition at line 88 of file tr2/type\_traits.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.1051 std::tr2::bool\_set Class Reference

### Public Member Functions

- constexpr [bool\\_set](#) ()
- constexpr [bool\\_set](#) (bool \_\_t)
- bool **contains** ([bool\\_set](#) \_\_b) const
- bool **equals** ([bool\\_set](#) \_\_b) const
- bool **is\_emptyset** () const
- bool **is\_indeterminate** () const
- bool **is\_singleton** () const
- **operator** bool () const

### Static Public Member Functions

- static [bool\\_set](#) **emptyset** ()
- static [bool\\_set](#) **indeterminate** ()

### Friends

- [bool\\_set](#) **operator!** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator&** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT , typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_out, [bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator==** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT , typename Traits >  
[std::basic\\_istream](#)< CharT, Traits > & **operator**>> ([std::basic\\_istream](#)< CharT, Traits > &\_\_in, [bool\\_set](#) &\_\_b)
- [bool\\_set](#) **operator^** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)

### 5.1051.1 Detailed Description

bool\_set

See N2136, Bool\_set: multi-valued logic by Hervé Brönnimann, Guillaume Melquiond, Sylvain Pion.

The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

Definition at line 54 of file bool\_set.

### 5.1051.2 Constructor & Destructor Documentation

#### 5.1051.2.1 bool\_set() [1/2]

```
constexpr std::tr2::bool_set::bool_set ( ) [inline]
```

Default constructor.

Definition at line 59 of file bool\_set.

#### 5.1051.2.2 bool\_set() [2/2]

```
constexpr std::tr2::bool_set::bool_set (
    bool __t ) [inline]
```

Constructor from bool.

Definition at line 62 of file bool\_set.

### 5.1051.3 Member Function Documentation

#### 5.1051.3.1 equals()

```
bool std::tr2::bool_set::equals (
    bool_set __b ) const [inline]
```

Return true if states are equal.

Definition at line 69 of file bool\_set.

### 5.1051.3.2 `is_emptyset()`

```
bool std::tr2::bool_set::is_emptyset ( ) const [inline]
```

Return true if this is empty.

Definition at line 73 of file `bool_set`.

### 5.1051.3.3 `is_indeterminate()`

```
bool std::tr2::bool_set::is_indeterminate ( ) const [inline]
```

Return true if this is indeterminate.

Definition at line 77 of file `bool_set`.

### 5.1051.3.4 `is_singleton()`

```
bool std::tr2::bool_set::is_singleton ( ) const [inline]
```

Return true if this is false or true (normal boolean).

Definition at line 81 of file `bool_set`.

### 5.1051.3.5 `operator bool()`

```
std::tr2::bool_set::operator bool ( ) const [inline]
```

Conversion to bool.

Definition at line 86 of file `bool_set`.

The documentation for this class was generated from the following files:

- [bool\\_set](#)
- [bool\\_set.tcc](#)

## 5.1052 `std::tr2::direct_bases<_Tp>` Struct Template Reference

### Public Types

- typedef [\\_\\_reflection\\_typelist](#)< `__direct_bases(_Tp)...` > **type**

## 5.1052.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::direct_bases< _Tp >
```

Enumerate all the direct base classes of a class. Form of a typelist.

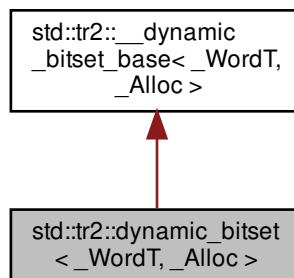
Definition at line 95 of file tr2/type\_traits.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 5.1053 std::tr2::dynamic\_bitset&lt; \_WordT, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::tr2::dynamic\_bitset< \_WordT, \_Alloc >:



## Classes

- class [reference](#)

## Public Types

- typedef `__dynamic_bitset_base< _WordT, _Alloc >` **\_Base**
- typedef `_Alloc` **allocator\_type**
- typedef `_WordT` **block\_type**
- typedef `bool` **const\_reference**
- typedef `size_t` **size\_type**



## Public Member Functions

- `dynamic_bitset` ()=default
- `dynamic_bitset` (const allocator\_type &\_\_alloc)
- `dynamic_bitset` (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type(), type())
- **dynamic\_bitset** (initializer\_list< block\_type > \_\_il, const allocator\_type &\_\_alloc=allocator\_type())
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
`dynamic_bitset` (const std::basic\_string< \_CharT, \_Traits, \_Alloc1 > &\_\_str, typename basic\_string< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_pos=0, typename basic\_string< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_n=std::basic\_string< \_CharT, \_Traits, \_Alloc1 >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'), const allocator\_type &\_\_alloc=allocator\_type())
- `dynamic_bitset` (const char \*\_\_str, const allocator\_type &\_\_alloc=allocator\_type())
- `dynamic_bitset` (const dynamic\_bitset &)=default
- `dynamic_bitset` (dynamic\_bitset &&\_\_b) noexcept
- template<typename \_Traits = std::char\_traits<char>, typename \_CharT = typename \_Traits::char\_type>  
void **\_M\_copy\_from\_ptr** (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
void **\_M\_copy\_from\_string** (const basic\_string< \_CharT, \_Traits, \_Alloc1 > &\_\_str, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
void **\_M\_copy\_to\_string** (std::basic\_string< \_CharT, \_Traits, \_Alloc1 > &\_\_str, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- bool `all` () const
- bool `any` () const
- void `append` (block\_type \_\_block)
- void **append** (initializer\_list< block\_type > \_\_il)
- template<typename \_BlockInputIterator >  
void `append` (\_BlockInputIterator \_\_first, \_BlockInputIterator \_\_last)
- void `clear` ()
- size\_type `count` () const noexcept
- bool `empty` () const noexcept
- size\_type `find_first` () const
- size\_type `find_next` (size\_t \_\_prev) const
- `dynamic_bitset` & `flip` ()
- `dynamic_bitset` & `flip` (size\_type \_\_pos)
- allocator\_type `get_allocator` () const noexcept
- bool **is\_proper\_subset\_of** (const dynamic\_bitset &\_\_b) const
- bool **is\_subset\_of** (const dynamic\_bitset &\_\_b) const
- constexpr size\_type `max_size` () noexcept
- bool `none` () const
- size\_type `num_blocks` () const noexcept
- `dynamic_bitset` & `operator=` (const dynamic\_bitset &)=default
- `dynamic_bitset` & `operator=` (dynamic\_bitset &&\_\_b) noexcept(std::is\_nothrow\_move\_assignable< \_Base >::value)
- `dynamic_bitset` `operator~` () const
- void `push_back` (bool \_\_bit)
- `dynamic_bitset` & `reset` ()
- `dynamic_bitset` & `reset` (size\_type \_\_pos)
- void `resize` (size\_type \_\_nbits, bool \_\_value=false)
- `dynamic_bitset` & `set` ()

- [dynamic\\_bitset](#) & [set](#) (size\_type \_\_pos, bool \_\_val=true)
- size\_type [size](#) () const noexcept
- void [swap](#) (dynamic\_bitset & \_\_b) noexcept
- bool [test](#) (size\_type \_\_pos) const
- template<typename \_CharT = char, typename \_Traits = std::char\_traits<\_CharT>, typename \_Alloc1 = std::allocator<\_CharT>>  
std::basic\_string<\_CharT, \_Traits, \_Alloc1 > [to\\_string](#) (\_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))  
const
- unsigned long long [to\\_ullong](#) () const
- unsigned long [to\\_ulong](#) () const

- [dynamic\\_bitset](#) & [operator&=](#) (const [dynamic\\_bitset](#) & \_\_rhs)
- [dynamic\\_bitset](#) & [operator&=](#) ([dynamic\\_bitset](#) && \_\_rhs)
- [dynamic\\_bitset](#) & [operator|=](#) (const [dynamic\\_bitset](#) & \_\_rhs)
- [dynamic\\_bitset](#) & [operator^=](#) (const [dynamic\\_bitset](#) & \_\_rhs)
- [dynamic\\_bitset](#) & [operator-=](#) (const [dynamic\\_bitset](#) & \_\_rhs)

- [dynamic\\_bitset](#) & [operator<<=](#) (size\_type \_\_pos)
- [dynamic\\_bitset](#) & [operator>>=](#) (size\_type \_\_pos)

- [reference operator\[\]](#) (size\_type \_\_pos)
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const

- [dynamic\\_bitset operator<<](#) (size\_type \_\_pos) const
- [dynamic\\_bitset operator>>](#) (size\_type \_\_pos) const

#### Static Public Attributes

- static const size\_type **bits\_per\_block**
- static const size\_type **npos**

### Private Member Functions

- `size_t _M_are_all_aux ()` const noexcept
- `void _M_clear ()` noexcept
- `void _M_do_and (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_append_block (block_type __block, size_type __pos)`
- `size_t _M_do_count ()` const noexcept
- `void _M_do_dif (const __dynamic_bitset_base &__x)` noexcept
- `size_type _M_do_find_first (size_t __not_found)` const
- `size_type _M_do_find_next (size_t __prev, size_t __not_found)` const
- `void _M_do_flip ()` noexcept
- `void _M_do_left_shift (size_t __shift)`
- `void _M_do_or (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_reset ()` noexcept
- `void _M_do_right_shift (size_t __shift)`
- `void _M_do_set ()` noexcept
- `unsigned long long _M_do_to_ullong ()` const
- `unsigned long _M_do_to_ulong ()` const
- `void _M_do_xor (const __dynamic_bitset_base &__x)` noexcept
- `allocator_type _M_get_allocator ()` const noexcept
- `block_type & _M_getword (size_type __pos)` noexcept
- `block_type _M_getword (size_type __pos)` const noexcept
- `block_type & _M_hiword ()` noexcept
- `block_type _M_hiword ()` const noexcept
- `bool _M_is_any ()` const noexcept
- `bool _M_is_equal (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_less (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_proper_subset_of (const __dynamic_bitset_base &__b)` const noexcept
- `bool _M_is_subset_of (const __dynamic_bitset_base &__b)` noexcept
- `void _M_resize (size_t __nbits, bool __value)`
- `size_type _M_size ()` const noexcept
- `void _M_swap (__dynamic_bitset_base &__b)` noexcept

### Static Private Member Functions

- `static block_type _S_maskbit (size_type __pos)` noexcept
- `static size_type _S_whichbit (size_type __pos)` noexcept
- `static size_type _S_whichbyte (size_type __pos)` noexcept
- `static size_type _S_whichword (size_type __pos)` noexcept

### Private Attributes

- `std::vector< block_type, allocator_type > _M_w`

### Static Private Attributes

- `static const size_type _S_bits_per_block`

## Friends

- bool **operator**< (const [dynamic\\_bitset](#) &\_\_lhs, const [dynamic\\_bitset](#) &\_\_rhs) noexcept
- bool **operator**== (const [dynamic\\_bitset](#) &\_\_lhs, const [dynamic\\_bitset](#) &\_\_rhs) noexcept
- class **reference**

## 5.1053.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >
```

The `dynamic_bitset` class represents a sequence of bits.

See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2050.pdf>

In the general unoptimized case, storage is allocated in word-sized blocks. Let  $B$  be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage.  $B - NbB$  bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `dynamic_bitset` as "a simple array of bits," be aware that your mental picture is reversed: a `dynamic_bitset` behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and the bit at index  $Nb-1$  in the "most significant / left-hand" position. Thus, unlike other containers, a `dynamic_bitset`'s index "counts from right to left," to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
    long a = 'a';
    dynamic_bitset<> b(a);
    cout << "b('a') is " << b << endl;
    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Most of the actual code isn't contained in `dynamic_bitset<>` itself, but in the base class `__dynamic_bitset_base`. The base class works with whole words, not with individual bits. This allows us to specialize `__dynamic_bitset_base` for the important special case where the `dynamic_bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `__dynamic_bitset_base` is a vector, and is indexed as such. This is carefully encapsulated.

Definition at line 419 of file `dynamic_bitset`.

## 5.1053.2 Constructor &amp; Destructor Documentation

**5.1053.2.1 dynamic\_bitset()** [1/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset ( ) [default]
```

All bits set to zero.

**5.1053.2.2 dynamic\_bitset()** [2/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    const allocator_type & __alloc ) [inline], [explicit]
```

All bits set to zero.

Definition at line 578 of file dynamic\_bitset.

**5.1053.2.3 dynamic\_bitset()** [3/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    size_type __nbits,
    unsigned long long __val = 0ULL,
    const allocator_type & __alloc = allocator_type() ) [inline], [explicit]
```

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 584 of file dynamic\_bitset.

**5.1053.2.4 dynamic\_bitset()** [4/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT , typename _Traits , typename _Alloc1 >
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    const std::basic_string< _CharT, _Traits, _Alloc1 > & __str,
    typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = 0,
    typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<↵
    _CharT, _Traits, _Alloc1>::npos,
    _CharT __zero = _CharT('0'),
    _CharT __one = _CharT('1'),
    const allocator_type & __alloc = allocator_type() ) [inline], [explicit]
```

Use a subset of a string.

## Parameters

<code>__str</code>	A string of '0' and '1' characters.
<code>__pos</code>	Index of the first character in <code>__str</code> to use.
<code>__n</code>	The number of characters to copy.
<code>__zero</code>	The character to use for unset bits.
<code>__one</code>	The character to use for set bits.
<code>__alloc</code>	An allocator.

## Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is bigger the size of <code>__str</code> .
<code>std::invalid_argument</code>	If a character appears in the string which is neither '0' nor '1'.

Definition at line 609 of file `dynamic_bitset`.

5.1053.2.5 `dynamic_bitset()` [5/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset<_WordT, _Alloc>::dynamic_bitset (
    const char * __str,
    const allocator_type & __alloc = allocator_type() ) [inline], [explicit]
```

Construct from a string.

## Parameters

<code>__str</code>	A string of '0' and '1' characters.
<code>__alloc</code>	An allocator.

## Exceptions

<code>std::invalid_argument</code>	If a character appears in the string which is neither '0' nor '1'.
------------------------------------	--

Definition at line 636 of file `dynamic_bitset`.

5.1053.2.6 `dynamic_bitset()` [6/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset<_WordT, _Alloc>::dynamic_bitset (
    const dynamic_bitset<_WordT, _Alloc> & ) [default]
```

Copy constructor.

#### 5.1053.2.7 `dynamic_bitset()` [7/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
    dynamic_bitset< _WordT, _Alloc > && __b ) [inline], [noexcept]
```

Move constructor.

Definition at line 648 of file `dynamic_bitset`.

### 5.1053.3 Member Function Documentation

#### 5.1053.3.1 `all()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all ( ) const [inline]
```

Tests whether all the bits are on.

##### Returns

True if all the bits are set.

Definition at line 1035 of file `dynamic_bitset`.

#### 5.1053.3.2 `any()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any ( ) const [inline]
```

Tests whether any of the bits are on.

##### Returns

True if at least one bit is set.

Definition at line 1043 of file `dynamic_bitset`.

### 5.1053.3.3 append() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset<_WordT, _Alloc >::append (
    block_type __block ) [inline]
```

Append a block.

Definition at line 725 of file dynamic\_bitset.

### 5.1053.3.4 append() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _BlockInputIterator >
void std::tr2::dynamic_bitset<_WordT, _Alloc >::append (
    _BlockInputIterator __first,
    _BlockInputIterator __last ) [inline]
```

Append an iterator range of blocks.

Definition at line 743 of file dynamic\_bitset.

### 5.1053.3.5 clear()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset<_WordT, _Alloc >::clear ( ) [inline]
```

Clear the bitset.

Definition at line 701 of file dynamic\_bitset.

### 5.1053.3.6 count()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc >::count ( ) const [inline], [noexcept]
```

Returns the number of bits which are set.

Definition at line 991 of file dynamic\_bitset.



### 5.1053.3.7 empty()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the dynamic\_bitset is empty.

Definition at line 1006 of file dynamic\_bitset.

### 5.1053.3.8 find\_first()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_first ( ) const [inline]
```

Finds the index of the first "on" bit.

#### Returns

The index of the first bit set, or size() if not found.

#### See also

find\_next

Definition at line 1071 of file dynamic\_bitset.

### 5.1053.3.9 find\_next()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_next (
    size_t __prev ) const [inline]
```

Finds the index of the next "on" bit after prev.

#### Returns

The index of the next bit set, or size() if not found.

#### Parameters

<code>__prev</code>	Where to start searching.
---------------------	---------------------------

See also

find\_first

Definition at line 1081 of file dynamic\_bitset.

#### 5.1053.3.10 flip() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::flip ( ) [inline]
```

Toggles every bit to its opposite value.

Definition at line 882 of file dynamic\_bitset.

#### 5.1053.3.11 flip() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::flip (
    size_type __pos ) [inline]
```

Toggles a given bit to its opposite value.

##### Parameters

<code>__pos</code>	The index of the bit.
--------------------	-----------------------

##### Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is bigger the size of the set.
--------------------------------	--

Definition at line 895 of file dynamic\_bitset.

#### 5.1053.3.12 get\_allocator()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
allocator_type std::tr2::dynamic_bitset<_WordT, _Alloc >::get_allocator ( ) const [inline],
[noexcept]
```

Return the allocator for the bitset.

Definition at line 681 of file dynamic\_bitset.

**5.1053.3.13** `max_size()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
constexpr size_type std::tr2::dynamic\_bitset< _WordT, _Alloc >::max_size ( ) [inline], [noexcept]
```

Returns the maximum size of a `dynamic_bitset` object having the same type as `*this`. The real answer is `max() * bits_per_block` but is likely to overflow.

Definition at line 1013 of file `dynamic_bitset`.

**5.1053.3.14** `none()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic\_bitset< _WordT, _Alloc >::none ( ) const [inline]
```

Tests whether any of the bits are on.

**Returns**

True if none of the bits are set.

Definition at line 1051 of file `dynamic_bitset`.

**5.1053.3.15** `num_blocks()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic\_bitset< _WordT, _Alloc >::num_blocks ( ) const [inline], [noexcept]
```

Returns the total number of blocks.

Definition at line 1001 of file `dynamic_bitset`.

**5.1053.3.16** `operator&=()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic\_bitset& std::tr2::dynamic\_bitset< _WordT, _Alloc >::operator&= (
    const dynamic\_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

Operations on `dynamic_bitsets`.

**Parameters**

<code>__rhs</code>	A same-sized <code>dynamic_bitset</code> .
--------------------	--

These should be self-explanatory.

Definition at line 758 of file dynamic\_bitset.

#### 5.1053.3.17 operator&=() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator&= (
    dynamic_bitset<_WordT, _Alloc > && __rhs ) [inline]
```

Operations on dynamic\_bitsets.

##### Parameters

<code>__rhs</code>	A same-sized dynamic_bitset.
--------------------	------------------------------

These should be self-explanatory.

Definition at line 765 of file dynamic\_bitset.

#### 5.1053.3.18 operator-=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator-= (
    const dynamic_bitset<_WordT, _Alloc > & __rhs ) [inline]
```

Operations on dynamic\_bitsets.

##### Parameters

<code>__rhs</code>	A same-sized dynamic_bitset.
--------------------	------------------------------

These should be self-explanatory.

Definition at line 786 of file dynamic\_bitset.

#### 5.1053.3.19 operator<<()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset<_WordT, _Alloc >::operator<< (
    size_type __pos ) const [inline]
```

Self-explanatory.

Definition at line 1057 of file dynamic\_bitset.

**5.1053.3.20 operator<<=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (
    size_type __pos ) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

<code>__pos</code>	The number of places to shift.
--------------------	--------------------------------

These should be self-explanatory.

Definition at line 801 of file dynamic\_bitset.

**5.1053.3.21 operator=() [1/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
    const dynamic_bitset< _WordT, _Alloc > & ) [default]
```

Copy assignment operator.

**5.1053.3.22 operator=() [2/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
    dynamic_bitset< _WordT, _Alloc > && __b ) [inline], [noexcept]
```

Move assignment operator.

Definition at line 665 of file dynamic\_bitset.

**5.1053.3.23 operator>>()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (
    size_type __pos ) const [inline]
```

Self-explanatory.

Definition at line 1061 of file dynamic\_bitset.

**5.1053.3.24 operator>>=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (
    size_type __pos ) [inline]
```

Operations on dynamic\_bitsets.

## Parameters

<code>__pos</code>	The number of places to shift.
--------------------	--------------------------------

These should be self-explanatory.

Definition at line 814 of file dynamic\_bitset.

## 5.1053.3.25 operator[]() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
    size_type __pos ) [inline]
```

Array-indexing support.

## Parameters

<code>__pos</code>	Index into the dynamic_bitset.
--------------------	--------------------------------

## Returns

A bool for a 'const dynamic\_bitset'. For non-const bitsets, an instance of the reference proxy class.

## Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 917 of file dynamic\_bitset.

## 5.1053.3.26 operator[]() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
const_reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
    size_type __pos ) const [inline]
```

Array-indexing support.

## Parameters

<code>__pos</code>	Index into the dynamic_bitset.
--------------------	--------------------------------

**Returns**

A bool for a 'const dynamic\_bitset'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 921 of file dynamic\_bitset.

**5.1053.3.27 operator^=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator^= (
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

<code>__rhs</code>	A same-sized dynamic_bitset.
--------------------	------------------------------

These should be self-explanatory.

Definition at line 779 of file dynamic\_bitset.

**5.1053.3.28 operator" |=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator|= (
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

<code>__rhs</code>	A same-sized dynamic_bitset.
--------------------	------------------------------

These should be self-explanatory.

Definition at line 772 of file dynamic\_bitset.

## 5.1053.3.29 operator~()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset< std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ ( ) const [inline]
```

See the no-argument flip().

Definition at line 904 of file dynamic\_bitset.

## 5.1053.3.30 push\_back()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (
    bool __bit ) [inline]
```

Push a bit onto the high end of the bitset.

Definition at line 711 of file dynamic\_bitset.

## 5.1053.3.31 reset() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset ( ) [inline]
```

Sets every bit to false.

Definition at line 857 of file dynamic\_bitset.

## 5.1053.3.32 reset() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (
    size_type __pos ) [inline]
```

Sets a given bit to false.

## Parameters

<code>__pos</code>	The index of the bit.
--------------------	-----------------------

## Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is bigger the size of the set.
--------------------------------	--



Same as writing `set (__pos, false)`.

Definition at line 871 of file `dynamic_bitset`.

#### 5.1053.3.33 `resize()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::resize (
    size_type __nbits,
    bool __value = false ) [inline]
```

Resize the bitset.

Definition at line 688 of file `dynamic_bitset`.

#### 5.1053.3.34 `set()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::set ( ) [inline]
```

Sets every bit to true.

Definition at line 832 of file `dynamic_bitset`.

#### 5.1053.3.35 `set()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::set (
    size_type __pos,
    bool __val = true ) [inline]
```

Sets a given bit to a particular value.

##### Parameters

<code>__pos</code>	The index of the bit.
<code>__val</code>	Either true or false, defaults to true.

##### Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is bigger the size of the set.
--------------------------------	--

Definition at line 846 of file `dynamic_bitset`.

**5.1053.3.36 size()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the total number of bits.

Definition at line 996 of file dynamic\_bitset.

Referenced by std::tr2::operator>>().

**5.1053.3.37 swap()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset<_WordT, _Alloc >::swap (
    dynamic_bitset<_WordT, _Alloc > & __b ) [inline], [noexcept]
```

Swap with another bitset.

Definition at line 654 of file dynamic\_bitset.

**5.1053.3.38 test()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset<_WordT, _Alloc >::test (
    size_type __pos ) const [inline]
```

Tests the value of a bit.

**Parameters**

<code>__pos</code>	The index of a bit.
--------------------	---------------------

**Returns**

The value at `__pos`.

**Exceptions**

<code>std::out_of_range</code>	If <code>__pos</code> is bigger the size of the set.
--------------------------------	--

Definition at line 1023 of file dynamic\_bitset.

**5.1053.3.39 to\_string()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 =
std::allocator<_CharT>>
std::basic_string<_CharT, _Traits, _Alloc1> std::tr2::dynamic_bitset< _WordT, _Alloc >::to_↵
string (
    _CharT __zero = _CharT('0'),
    _CharT __one = _CharT('1') ) const [inline]
```

Returns a character interpretation of the `dynamic_bitset`.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 957 of file `dynamic_bitset`.

**5.1053.3.40 to\_ullong()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ullong ( ) const [inline]
```

Returns a numerical interpretation of the `dynamic_bitset`.

**Returns**

The integral equivalent of the bits.

**Exceptions**

<code>std::overflow_error</code>	If there are too many bits to be represented in an unsigned long.
----------------------------------	---

Definition at line 942 of file `dynamic_bitset`.

**5.1053.3.41 to\_ulong()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ulong ( ) const [inline]
```

Returns a numerical interpretation of the `dynamic_bitset`.

## Returns

The integral equivalent of the bits.

## Exceptions

<code>std::overflow_error</code>	If there are too many bits to be represented in an <code>unsigned long</code> .
----------------------------------	---

Definition at line 932 of file `dynamic_bitset`.

The documentation for this class was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

5.1054 `std::tr2::dynamic_bitset<_WordT, _Alloc>::reference` Class Reference

## Public Member Functions

- **reference** ([dynamic\\_bitset](#) &\_\_b, size\_type \_\_pos) noexcept
- [reference](#) & **flip** () noexcept
- **operator bool** () const noexcept
- [reference](#) & **operator=** (bool \_\_x) noexcept
- [reference](#) & **operator=** (const [reference](#) &\_\_j) noexcept
- bool **operator~** () const noexcept

## Friends

- class **dynamic\_bitset**

## 5.1054.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset<_WordT, _Alloc>::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)

Definition at line 513 of file `dynamic_bitset`.

The documentation for this class was generated from the following file:

- [dynamic\\_bitset](#)

## 5.1055 std::try\_to\_lock\_t Struct Reference

### 5.1055.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

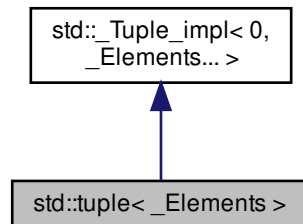
Definition at line 135 of file std\_mutex.h.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 5.1056 std::tuple<\_Elements> Class Template Reference

Inheritance diagram for std::tuple<\_Elements>:



### Public Types

- `template<typename _Dummy>`  
`using _TCC = _TC<is_same<_Dummy, void>::value, _Elements...>`
- `template<typename... _UElements>`  
`using _TMC = _TC<(sizeof...(_Elements)==sizeof...(_UElements)) &&(_TC<(sizeof...(_UElements)==1), _Elements...>::template _NotSameTuple<_UElements...>()), _Elements...>`
- `template<typename... _UElements>`  
`using _TMCT = _TC<(sizeof...(_Elements)==sizeof...(_UElements)) &&is_same<tuple<_Elements...>, tuple<_UElements...>::value, _Elements...>`
- `template<typename _Dummy>`  
`using _TNTC = _TC<is_same<_Dummy, void>::value &&sizeof...(_Elements)==1, _Elements...>`

## Public Member Functions

- template<typename \_Dummy = void, typename enable\_if< \_TCC< \_Dummy >::template \_ConstructibleTuple< \_Elements... >() &&!\_TCC< \_Dummy >::template \_ImplicitlyConvertibleTuple< \_Elements... >() &&(sizeof...( \_Elements ) >= 1)>  
constexpr **tuple** (const \_Elements &... \_\_elements)
- template<typename \_Dummy = void, typename enable\_if< \_TCC< \_Dummy >::template \_ConstructibleTuple< \_Elements... >() &&!\_TCC< \_Dummy >::template \_ImplicitlyConvertibleTuple< \_Elements... >() &&(sizeof...( \_Elements ) >= 1)>  
constexpr **tuple** (const \_Elements &... \_\_elements)
- template<typename... \_UElements, typename enable\_if< \_TMC< \_UElements... >::template \_MoveConstructibleTuple< \_UElements... >() &&\_TMC< \_UElements... >::template \_ImplicitlyMoveConvertibleTuple< \_UElements... >() &&(sizeof...( \_Elements ) >= 1)>  
constexpr **tuple** ( \_UElements &&... \_\_elements)
- template<typename... \_UElements, typename enable\_if< \_TMC< \_UElements... >::template \_MoveConstructibleTuple< \_UElements... >() &&\_TMC< \_UElements... >::template \_ImplicitlyMoveConvertibleTuple< \_UElements... >() &&(sizeof...( \_Elements ) >= 1)>  
constexpr **tuple** ( \_UElements &&... \_\_elements)
- constexpr **tuple** (const **tuple** &)=default
- constexpr **tuple** (**tuple** &&)=default
- template<typename... \_UElements, typename \_Dummy = void, typename enable\_if< \_TMCT< \_UElements... >::template \_ConstructibleTuple< \_UElements... >() &&\_TMCT< \_UElements... >::template \_ImplicitlyConvertibleTuple< \_UElements... >() &&\_TNTC< \_Dummy >::template \_NonNestedTuple< const tuple< \_UElements... > &>(), bool >::type = true>  
constexpr **tuple** (const **tuple**< \_UElements... > &\_\_in)
- template<typename... \_UElements, typename \_Dummy = void, typename enable\_if< \_TMCT< \_UElements... >::template \_ConstructibleTuple< \_UElements... >() &&!\_TMCT< \_UElements... >::template \_ImplicitlyConvertibleTuple< \_UElements... >() &&\_TNTC< \_Dummy >::template \_NonNestedTuple< const tuple< \_UElements... > &>(), bool >::type = false>  
constexpr **tuple** (const **tuple**< \_UElements... > &\_\_in)
- template<typename... \_UElements, typename \_Dummy = void, typename enable\_if< \_TMCT< \_UElements... >::template \_MoveConstructibleTuple< \_UElements... >() &&\_TMCT< \_UElements... >::template \_ImplicitlyMoveConvertibleTuple< \_UElements... >() &&\_TNTC< \_Dummy >::template \_NonNestedTuple< tuple< \_UElements... > &&>(), bool >::type = true>  
constexpr **tuple** (**tuple**< \_UElements... > &&\_\_in)
- template<typename... \_UElements, typename \_Dummy = void, typename enable\_if< \_TMCT< \_UElements... >::template \_MoveConstructibleTuple< \_UElements... >() &&!\_TMCT< \_UElements... >::template \_ImplicitlyMoveConvertibleTuple< \_UElements... >() &&\_TNTC< \_Dummy >::template \_NonNestedTuple< tuple< \_UElements... > &&>(), bool >::type = false>  
constexpr **tuple** (**tuple**< \_UElements... > &&\_\_in)
- template<typename \_Alloc >  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a)
- template<typename \_Alloc , typename \_Dummy = void, typename enable\_if< \_TCC< \_Dummy >::template \_ConstructibleTuple< \_Elements... >() &&\_TCC< \_Dummy >::template \_ImplicitlyConvertibleTuple< \_Elements... >(), bool >::type = true>  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, const \_Elements &... \_\_elements)
- template<typename \_Alloc , typename \_Dummy = void, typename enable\_if< \_TCC< \_Dummy >::template \_ConstructibleTuple< \_Elements... >() &&!\_TCC< \_Dummy >::template \_ImplicitlyConvertibleTuple< \_Elements... >(), bool >::type = false>  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, const \_Elements &... \_\_elements)
- template<typename \_Alloc , typename... \_UElements, typename enable\_if< \_TMC< \_UElements... >::template \_MoveConstructibleTuple< \_UElements... >() &&\_TMC< \_UElements... >::template \_ImplicitlyMoveConvertibleTuple< \_UElements... >(), bool >::type = true>  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, \_UElements &&... \_\_elements)
- template<typename \_Alloc , typename... \_UElements, typename enable\_if< \_TMC< \_UElements... >::template \_MoveConstructibleTuple< \_UElements... >() &&!\_TMC< \_UElements... >::template \_ImplicitlyMoveConvertibleTuple< \_UElements... >(), bool >::type = false>  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, \_UElements &&... \_\_elements)
- template<typename \_Alloc >  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, const **tuple** &\_\_in)
- template<typename \_Alloc >  
**tuple** (**allocator\_arg\_t** \_\_tag, const \_Alloc &\_\_a, **tuple** &&\_\_in)

- `template<typename _Alloc , typename _Dummy = void, typename... _UElements, typename enable_if< _TMCT< _UElements... >↔  
::template _ConstructibleTuple< _UElements... >() &&_TMCT< _UElements... >::template _ImplicitlyConvertibleTuple< _UElements...  
>() &&_TNTC< _Dummy >::template _NonNestedTuple< tuple< _UElements... > && >(), bool >::type = true>  
tuple (allocator\_arg\_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc , typename _Dummy = void, typename... _UElements, typename enable_if< _TMCT< _UElements... >↔  
::template _ConstructibleTuple< _UElements... >() &&!_TMCT< _UElements... >::template _ImplicitlyConvertibleTuple< _UElements...  
>() &&_TNTC< _Dummy >::template _NonNestedTuple< tuple< _UElements... > && >(), bool >::type = false>  
tuple (allocator\_arg\_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc , typename _Dummy = void, typename... _UElements, typename enable_if< _TMCT< _UElements... >↔  
::template _MoveConstructibleTuple< _UElements... >() &&_TMCT< _UElements... >::template _ImplicitlyMoveConvertibleTuple< _U↔  
Elements... >() &&_TNTC< _Dummy >::template _NonNestedTuple< tuple< _UElements... > && >(), bool >::type = true>  
tuple (allocator\_arg\_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `template<typename _Alloc , typename _Dummy = void, typename... _UElements, typename enable_if< _TMCT< _UElements... >↔  
::template _MoveConstructibleTuple< _UElements... >() &&!_TMCT< _UElements... >::template _ImplicitlyMoveConvertibleTuple< _U↔  
Elements... >() &&_TNTC< _Dummy >::template _NonNestedTuple< tuple< _UElements... > && >(), bool >::type = false>  
tuple (allocator\_arg\_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- [tuple](#) & **operator=** (const [tuple](#) &\_\_in)
- [tuple](#) & **operator=** ([tuple](#) &&\_\_in) noexcept([is\\_nothrow\\_move\\_assignable](#)< [\\_Inherited](#) >::value)
- `template<typename... _UElements>  
enable\_if< sizeof...( _UElements)==sizeof...( _Elements), tuple & >::type operator= (const tuple< _UElements...  
> &__in)`
- `template<typename... _UElements>  
enable\_if< sizeof...( _UElements)==sizeof...( _Elements), tuple & >::type operator= (tuple< _UElements... >  
&&__in)`
- `void swap (tuple &__in) noexcept(noexcept(__in._M_swap(__in)))`

### 5.1056.1 Detailed Description

```
template<typename... _Elements>
class std::tuple< _Elements >
```

Primary class template, [tuple](#).

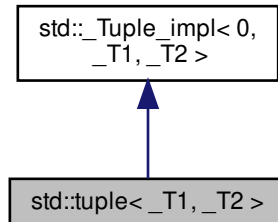
Definition at line 53 of file [tuple](#).

The documentation for this class was generated from the following file:

- [tuple](#)

## 5.1057 std::tuple&lt; \_T1, \_T2 &gt; Class Template Reference

Inheritance diagram for std::tuple< \_T1, \_T2 >:



## Public Types

- template<typename \_Dummy >  
using **\_TCC** = \_TC< [is\\_same](#)< \_Dummy, void >::value, \_T1, \_T2 >
- using **\_TMC** = \_TC< true, \_T1, \_T2 >

## Public Member Functions

- template<typename \_Dummy = void, typename enable\_if< \_TCC< \_Dummy >::template \_ConstructibleTuple< \_T1, \_T2 >() &&\_TCC< \_Dummy >::template \_ImplicitlyConvertibleTuple< \_T1, \_T2 >(), bool >::type = true>  
constexpr **tuple** (const \_T1 &\_\_a1, const \_T2 &\_\_a2)
- template<typename \_Dummy = void, typename enable\_if< \_TCC< \_Dummy >::template \_ConstructibleTuple< \_T1, \_T2 >() &&!\_TCC< \_Dummy >::template \_ImplicitlyConvertibleTuple< \_T1, \_T2 >(), bool >::type = false>  
constexpr **tuple** (const \_T1 &\_\_a1, const \_T2 &\_\_a2)
- template<typename \_U1 , typename \_U2 , typename enable\_if< \_TMC::template \_MoveConstructibleTuple< \_U1, \_U2 >() &&\_TMC< \_U1, \_U2 >::template \_ImplicitlyMoveConvertibleTuple< \_U1, \_U2 >() &&is\_same< typename decay< \_U1 >::type, allocator\_arg\_t >::value, bool >::type = true>  
constexpr **tuple** (\_U1 &&\_\_a1, \_U2 &&\_\_a2)
- template<typename \_U1 , typename \_U2 , typename enable\_if< \_TMC::template \_MoveConstructibleTuple< \_U1, \_U2 >() &&!\_TMC< \_U1, \_U2 >::template \_ImplicitlyMoveConvertibleTuple< \_U1, \_U2 >() &&is\_same< typename decay< \_U1 >::type, allocator\_arg\_t >::value, bool >::type = false>  
constexpr **tuple** (\_U1 &&\_\_a1, \_U2 &&\_\_a2)
- constexpr **tuple** (const [tuple](#) &)=default
- constexpr **tuple** ([tuple](#) &&)=default
- template<typename \_U1 , typename \_U2 , typename enable\_if< \_TMC::template \_ConstructibleTuple< \_U1, \_U2 >() &&\_TMC::template \_ImplicitlyConvertibleTuple< \_U1, \_U2 >(), bool >::type = true>  
constexpr **tuple** (const [tuple](#)< \_U1, \_U2 > &\_\_in)
- template<typename \_U1 , typename \_U2 , typename enable\_if< \_TMC::template \_ConstructibleTuple< \_U1, \_U2 >() &&!\_TMC::template \_ImplicitlyConvertibleTuple< \_U1, \_U2 >(), bool >::type = false>  
constexpr **tuple** (const [tuple](#)< \_U1, \_U2 > &\_\_in)





- `template<typename _Alloc, typename _U1, typename _U2, typename enable_if< _TMC::template _MoveConstructibleTuple< _U1, _U2 >() &&_TMC::template _ImplicitlyMoveConvertibleTuple< _U1, _U2 >(), bool >::type = true>`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2, typename enable_if< _TMC::template _MoveConstructibleTuple< _U1, _U2 >() &&!_TMC::template _ImplicitlyMoveConvertibleTuple< _U1, _U2 >(), bool >::type = false>`  
`tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in) noexcept(is_nothrow_move_assignable< _Inherited >::value)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (const tuple< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (const pair< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`  
`tuple & operator= (pair< _U1, _U2 > &&__in)`
- `void swap (tuple &__in) noexcept(noexcept(__in._M_swap(__in)))`

#### 5.1057.1 Detailed Description

```
template<typename _T1, typename _T2>
class std::tuple< _T1, _T2 >
```

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.

Definition at line 907 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 5.1058 std::tuple\_element< \_Int, \_Tp > Struct Template Reference

#### 5.1058.1 Detailed Description

```
template<std::size_t _Int, typename _Tp>
struct std::tuple_element< _Int, _Tp >
```

`tuple_element`

Gives the type of the ith element of a given tuple type.

Definition at line 359 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

### 5.1059 `std::tuple_element< 0, std::pair< _Tp1, _Tp2 > >` Struct Template Reference

#### Public Types

- `typedef _Tp1 type`

#### 5.1059.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_element< 0, std::pair< _Tp1, _Tp2 > >
```

Partial specialization for `std::pair`.

Definition at line 155 of file `utility`.

The documentation for this struct was generated from the following file:

- [utility](#)

### 5.1060 `std::tuple_element< 0, tuple< _Head, _Tail... > >` Struct Template Reference

#### Public Types

- `typedef _Head type`

#### 5.1060.1 Detailed Description

```
template<typename _Head, typename... _Tail>
struct std::tuple_element< 0, tuple< _Head, _Tail... > >
```

Basis case for `tuple_element`: The first element is the one we're seeking.

Definition at line 1286 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

### 5.1061 `std::tuple_element< 1, std::pair< _Tp1, _Tp2 > >` Struct Template Reference

#### Public Types

- `typedef _Tp2 type`

## 5.1061.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_element< 1, std::pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.

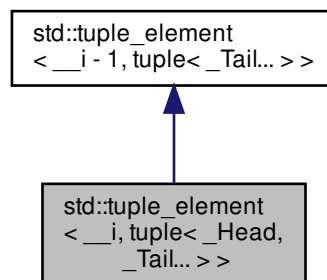
Definition at line 160 of file utility.

The documentation for this struct was generated from the following file:

- [utility](#)

## 5.1062 std::tuple\_element&lt; \_\_i, tuple&lt; \_Head, \_Tail... &gt; &gt; Struct Template Reference

Inheritance diagram for std::tuple\_element< \_\_i, tuple< \_Head, \_Tail... > >:



## 5.1062.1 Detailed Description

```
template<std::size_t __i, typename _Head, typename... _Tail>
struct std::tuple_element< __i, tuple< _Head, _Tail... > >
```

Recursive case for tuple\_element: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

Definition at line 1279 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

### 5.1063 `std::tuple_element< __i, tuple<> >` Struct Template Reference

#### 5.1063.1 Detailed Description

```
template<size_t __i>
struct std::tuple_element< __i, tuple<> >
```

Error case for `tuple_element`: invalid index.

Definition at line 1295 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

### 5.1064 `std::tuple_element< _Int, ::array< _Tp, _Nm > >` Struct Template Reference

#### Public Types

- `typedef _Tp type`

#### 5.1064.1 Detailed Description

```
template<std::size_t _Int, typename _Tp, std::size_t _Nm>
struct std::tuple_element< _Int, ::array< _Tp, _Nm > >
```

Partial specialization for `std::array`.

Definition at line 363 of file `array`.

The documentation for this struct was generated from the following file:

- [array](#)

### 5.1065 `std::tuple_element< _Int, std::__debug::array< _Tp, _Nm > >` Struct Template Reference

#### Public Types

- `typedef _Tp type`

## 5.1065.1 Detailed Description

```
template<std::size_t _Int, typename _Tp, std::size_t _Nm>
struct std::tuple_element< _Int, std::__debug::array< _Tp, _Nm > >
```

tuple\_element

Definition at line 329 of file debug/array.

The documentation for this struct was generated from the following file:

- [debug/array](#)

## 5.1066 std::tuple\_size&lt; \_Tp &gt; Struct Template Reference

Inherited by std::tuple\_size< const \_\_enable\_if\_has\_tuple\_size< \_Tp > >, std::tuple\_size< const volatile \_\_enable\_if\_has\_tuple\_size< \_Tp > >, and std::tuple\_size< volatile \_\_enable\_if\_has\_tuple\_size< \_Tp > >.

## 5.1066.1 Detailed Description

```
template<typename _Tp>
struct std::tuple_size< _Tp >
```

tuple\_size

Finds the size of a given tuple type.

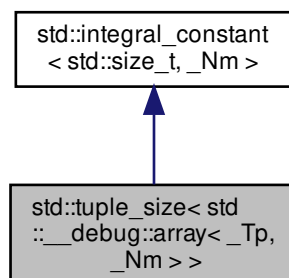
Definition at line 350 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

## 5.1067 std::tuple\_size&lt; std::\_\_debug::array&lt; \_Tp, \_Nm &gt; &gt; Struct Template Reference

Inheritance diagram for std::tuple\_size< std::\_\_debug::array< \_Tp, \_Nm > >:



#### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr std::size\_t **value**

#### 5.1067.1 Detailed Description

```
template<typename _Tp, std::size_t _Nm>
struct std::tuple_size< std::__debug::array< _Tp, _Nm > >
```

tuple\_size

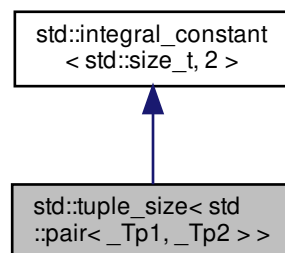
Definition at line 324 of file debug/array.

The documentation for this struct was generated from the following file:

- [debug/array](#)

#### 5.1068 std::tuple\_size< std::pair< \_Tp1, \_Tp2 > > Struct Template Reference

Inheritance diagram for std::tuple\_size< std::pair< \_Tp1, \_Tp2 > >:



## Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr std::size\_t **value**

## 5.1068.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_size< std::pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.

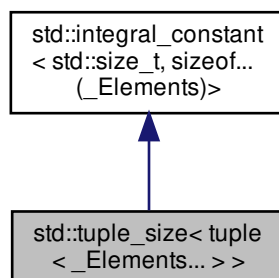
Definition at line 150 of file utility.

The documentation for this struct was generated from the following file:

- [utility](#)

## 5.1069 std::tuple\_size&lt; tuple&lt; \_Elements... &gt; &gt; Struct Template Reference

Inheritance diagram for std::tuple\_size< tuple< \_Elements... > >:





#### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr std::size\_t **value**

#### 5.1069.1 Detailed Description

```
template<typename... _Elements>
struct std::tuple_size< tuple< _Elements... > >
```

class tuple\_size

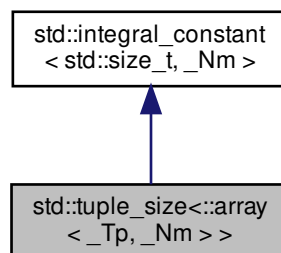
Definition at line 1266 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

#### 5.1070 std::tuple\_size<::array< \_Tp, \_Nm > > Struct Template Reference

Inheritance diagram for std::tuple\_size<::array< \_Tp, \_Nm > >:



### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr std::size\_t **value**

#### 5.1070.1 Detailed Description

```
template<typename _Tp, std::size_t _Nm>
struct std::tuple_size<::array< _Tp, _Nm > >
```

Partial specialization for std::array.

Definition at line 354 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

## 5.1071 std::type\_index Struct Reference

### Public Member Functions

- **type\_index** (const [type\\_info](#) &\_\_rhs) noexcept
- size\_t **hash\_code** () const noexcept
- const char \* **name** () const noexcept
- bool **operator!=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator==** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>=** (const [type\\_index](#) &\_\_rhs) const noexcept

### 5.1071.1 Detailed Description

#### Class `type_index`

The class `type_index` provides a simple wrapper for `type_info` which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Definition at line 52 of file `typeidindex`.

The documentation for this struct was generated from the following file:

- [typeidindex](#)

### 5.1072 `std::type_info` Class Reference

Inherited by `__cxxabiv1::__array_type_info`, `__cxxabiv1::__class_type_info`, `__cxxabiv1::__enum_type_info`, `__cxxabiv1::__function_type_info`, `__cxxabiv1::__fundamental_type_info`, and `__cxxabiv1::__pbase_type_info`.

#### Public Member Functions

- virtual `~type_info` ()
- virtual bool `__do_catch` (const `type_info` \* \_\_thr\_type, void \*\* \_\_thr\_obj, unsigned \_\_outer) const
- virtual bool `__do_upcast` (const `__cxxabiv1::__class_type_info` \* \_\_target, void \*\* \_\_obj\_ptr) const
- virtual bool `__is_function_p` () const
- virtual bool `__is_pointer_p` () const
- bool `before` (const `type_info` & \_\_arg) const noexcept
- size\_t `hash_code` () const noexcept
- const char \* `name` () const noexcept
- bool `operator!=` (const `type_info` & \_\_arg) const noexcept
- bool `operator==` (const `type_info` & \_\_arg) const noexcept

#### Protected Member Functions

- `type_info` (const char \* \_\_n)

#### Protected Attributes

- const char \* `__name`

### 5.1072.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

Definition at line 88 of file `typeinfo`.

## 5.1072.2 Constructor &amp; Destructor Documentation

## 5.1072.2.1 ~type\_info()

```
virtual std::type_info::~~type_info ( ) [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated type\_info structures in the new-abi.

## 5.1072.3 Member Function Documentation

## 5.1072.3.1 name()

```
const char* std::type_info::name ( ) const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!

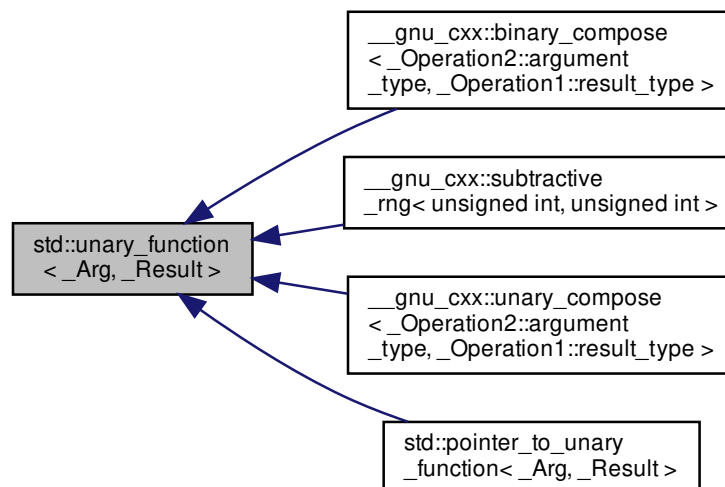
Definition at line 99 of file typeinfo.

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.1073 std::unary\_function&lt;\_Arg, \_Result&gt; Struct Template Reference

Inheritance diagram for std::unary\_function<\_Arg, \_Result>:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### 5.1073.1 Detailed Description

```
template<typename _Arg, typename _Result>
struct std::unary_function< _Arg, _Result >
```

This is one of the [functor base classes](#).

Definition at line 105 of file `stl_function.h`.

### 5.1073.2 Member Typedef Documentation

#### 5.1073.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary\_function< \_Arg, \_Result >::argument\_type
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 5.1073.2.2 `result_type`

```
template<typename _Arg, typename _Result>
typedef _Result std::unary\_function< \_Arg, \_Result >::result\_type
```

`result_type` is the return type

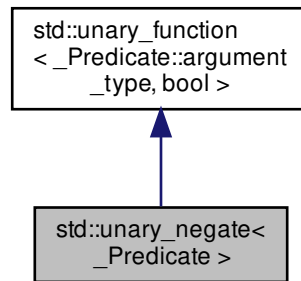
Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.1074 std::unary\_negate&lt; \_Predicate &gt; Class Template Reference

Inheritance diagram for std::unary\_negate< \_Predicate >:



## Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- `_GLIBCXX14_CONSTEXPR` **unary\_negate** (`const _Predicate &__x`)
- `_GLIBCXX14_CONSTEXPR` `bool` **operator()** (`const typename _Predicate::argument_type &__x`) `const`

## Protected Attributes

- `_Predicate` **\_M\_pred**

## 5.1074.1 Detailed Description

```
template<typename _Predicate>
class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 979 of file `stl_function.h`.

## 5.1074.2 Member Typedef Documentation

### 5.1074.2.1 `argument_type`

```
typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type , bool >::argument_type  
[inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

### 5.1074.2.2 `result_type`

```
typedef bool std::unary_function< _Predicate::argument_type , bool >::result_type [inherited]
```

`result_type` is the return type

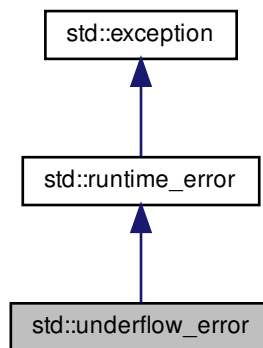
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.1075 `std::underflow_error` Class Reference

Inheritance diagram for `std::underflow_error`:



### Public Member Functions

- **`underflow_error`** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **`underflow_error`** (const char \*) \_GLIBCXX\_TXN\_SAFE
- virtual const char \* [what](#) () const \_GLIBCXX\_TXN\_SAFE\_DYN noexcept

## 5.1075.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 252 of file `stdexcept`.

## 5.1075.2 Member Function Documentation

5.1075.2.1 `what()`

```
virtual const char* std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.1076 `std::underlying_type<_Tp>` Struct Template Reference

## Public Member Functions

- `typedef __underlying_type (_Tp) type`

## 5.1076.1 Detailed Description

```
template<typename _Tp>  
struct std::underlying_type<_Tp>
```

The underlying type of an enum.

Definition at line 2000 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.1077 `std::uniform_int_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)



## Public Types

- typedef `_IntType` `result_type`

## Public Member Functions

- `uniform_int_distribution` (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType>::max()`)
- `uniform_int_distribution` (const `param_type` &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void `__generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void `__generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >  
void `__generate` (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `result_type a` () const
- `result_type b` () const
- `result_type max` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator` >  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `param_type param` () const
- void `param` (const `param_type` &\_\_param)
- void `reset` ()

## Friends

- bool `operator==` (const `uniform_int_distribution` &\_\_d1, const `uniform_int_distribution` &\_\_d2)

## 5.1077.1 Detailed Description

```
template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Definition at line 58 of file `uniform_int_dist.h`.

## 5.1077.2 Member Typedef Documentation

### 5.1077.2.1 `result_type`

```
template<typename _IntType = int>
typedef _IntType std::uniform_int_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 61 of file `uniform_int_dist.h`.

## 5.1077.3 Constructor & Destructor Documentation

### 5.1077.3.1 `uniform_int_distribution()`

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution (
    _IntType __a = 0,
    _IntType __b = std::numeric_limits<_IntType>::max() ) [inline], [explicit]
```

Constructs a uniform distribution object.

Definition at line 105 of file `uniform_int_dist.h`.

## 5.1077.4 Member Function Documentation

### 5.1077.4.1 `max()`

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::max ( ) const [inline]
```

Returns the inclusive upper bound of the distribution range.

Definition at line 157 of file `uniform_int_dist.h`.

### 5.1077.4.2 `min()`

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::min ( ) const [inline]
```

Returns the inclusive lower bound of the distribution range.

Definition at line 150 of file `uniform_int_dist.h`.

#### 5.1077.4.3 operator()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::uniform_int_distribution< _IntType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 165 of file uniform\_int\_dist.h.

#### 5.1077.4.4 param() [1/2]

```
template<typename _IntType = int>
param_type std::uniform_int_distribution< _IntType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 135 of file uniform\_int\_dist.h.

Referenced by std::operator>>().

#### 5.1077.4.5 param() [2/2]

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 143 of file uniform\_int\_dist.h.

#### 5.1077.4.6 reset()

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::reset ( ) [inline]
```

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 121 of file uniform\_int\_dist.h.

## 5.1077.5 Friends And Related Function Documentation

## 5.1077.5.1 operator==

```
template<typename _IntType = int>
bool operator== (
    const uniform_int_distribution< _IntType > & __d1,
    const uniform_int_distribution< _IntType > & __d2 ) [friend]
```

Return true if two uniform integer distributions have the same parameters.

Definition at line 200 of file uniform\_int\_dist.h.

The documentation for this class was generated from the following file:

- [uniform\\_int\\_dist.h](#)

## 5.1078 std::uniform\_int\_distribution&lt; \_IntType &gt;::param\_type Struct Reference

## Public Types

- typedef [uniform\\_int\\_distribution](#)< \_IntType > **distribution\_type**

## Public Member Functions

- **param\_type** ( \_IntType \_\_a=0, \_IntType \_\_b=[std::numeric\\_limits](#)< \_IntType >::max())
- **result\_type a** () const
- **result\_type b** () const

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 5.1078.1 Detailed Description

```
template<typename _IntType = int>
struct std::uniform_int_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 67 of file uniform\_int\_dist.h.

The documentation for this struct was generated from the following file:

- [uniform\\_int\\_dist.h](#)

## 5.1079 std::uniform\_real\_distribution< \_RealType > Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- [uniform\\_real\\_distribution](#) (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **uniform\_real\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **a** () const
- [result\\_type](#) **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

### Friends

- bool **operator==** (const [uniform\\_real\\_distribution](#) &\_\_d1, const [uniform\\_real\\_distribution](#) &\_\_d2)

### 5.1079.1 Detailed Description

```
template<typename _RealType = double>
class std::uniform_real_distribution< _RealType >
```

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1702 of file random.h.

## 5.1079.2 Member Typedef Documentation

5.1079.2.1 `result_type`

```
template<typename _RealType = double>
typedef _RealType std::uniform_real_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 1705 of file `random.h`.

## 5.1079.3 Constructor &amp; Destructor Documentation

5.1079.3.1 `uniform_real_distribution()`

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution (
    _RealType __a = _RealType(0),
    _RealType __b = _RealType(1) ) [inline], [explicit]
```

Constructs a `uniform_real_distribution` object.

## Parameters

<code>↔</code> __a	[IN] The lower bound of the distribution.
<code>↔</code> __b	[IN] The upper bound of the distribution.

Definition at line 1753 of file `random.h`.

## 5.1079.4 Member Function Documentation

5.1079.4.1 `max()`

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::max ( ) const [inline]
```

Returns the inclusive upper bound of the distribution range.

Definition at line 1805 of file `random.h`.

#### 5.1079.4.2 min()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::min ( ) const [inline]
```

Returns the inclusive lower bound of the distribution range.

Definition at line 1798 of file random.h.

#### 5.1079.4.3 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::uniform_real_distribution< _RealType >::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 1813 of file random.h.

#### 5.1079.4.4 param() [1/2]

```
template<typename _RealType = double>
param_type std::uniform_real_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 1783 of file random.h.

Referenced by std::operator>>().

#### 5.1079.4.5 param() [2/2]

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 1791 of file random.h.

#### 5.1079.4.6 `reset()`

```
template<typename _RealType = double>
void std::uniform_real_distribution<_RealType>::reset ( ) [inline]
```

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1769 of file `random.h`.

### 5.1079.5 Friends And Related Function Documentation

#### 5.1079.5.1 `operator==`

```
template<typename _RealType = double>
bool operator== (
    const uniform_real_distribution<_RealType> & __d1,
    const uniform_real_distribution<_RealType> & __d2 ) [friend]
```

Return true if two uniform real distributions have the same parameters.

Definition at line 1853 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 5.1080 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

#### Public Types

- typedef `uniform_real_distribution<_RealType>` **distribution\_type**

#### Public Member Functions

- **param\_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- **result\_type a** () const
- **result\_type b** () const



## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.1080.1 Detailed Description

```
template<typename _RealType = double>
struct std::uniform_real_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 1712 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.1081 `std::unique_lock< _Mutex >` Class Template Reference

#### Public Types

- typedef `_Mutex` **mutex\_type**

#### Public Member Functions

- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- template<typename \_Clock, typename \_Duration >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Rep, typename \_Period >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- **unique\_lock** (const [unique\\_lock](#) &)=delete
- **unique\_lock** ([unique\\_lock](#) &&\_\_u) noexcept
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- [unique\\_lock](#) & **operator=** (const [unique\\_lock](#) &)=delete
- [unique\\_lock](#) & **operator=** ([unique\\_lock](#) &&\_\_u) noexcept
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** ([unique\\_lock](#) &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

## 5.1081.1 Detailed Description

```
template<typename _Mutex>
class std::unique_lock< _Mutex >
```

A movable scoped lock type.

A `unique_lock` controls mutex ownership within a scope. Ownership of the mutex can be delayed until after construction and can be transferred to another `unique_lock` by move construction or move assignment. If a mutex lock is owned when the destructor runs ownership will be released.

Definition at line 185 of file `std_mutex.h`.

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

## 5.1082 std::unique\_ptr&lt; \_Tp, \_Dp &gt; Class Template Reference

## Public Types

- `template<typename _Up, typename _Ep >`  
`using __safe_conversion_up = __and_< is_convertible< typename unique_ptr< _Up, _Ep >::pointer, pointer >, __not_< is_array< _Up > > >`
- `using deleter_type = _Dp`
- `using element_type = _Tp`
- `using pointer = typename __uniq_ptr_impl< _Tp, _Dp >::pointer`

## Public Member Functions

- `template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>`  
`constexpr unique_ptr () noexcept`
- `template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>`  
`unique_ptr (pointer __p) noexcept`
- `unique_ptr (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) noexcept`
- `unique_ptr (pointer __p, typename remove_reference< deleter_type >::type &&__d) noexcept`
- `template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>`  
`constexpr unique_ptr (nullptr_t) noexcept`
- `unique_ptr (unique_ptr &&__u) noexcept`
- `template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, typename conditional<is_↔reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>::type>>`  
`unique_ptr (unique_ptr< _Up, _Ep > &&__u) noexcept`
- `template<typename _Up, typename >`  
`unique_ptr (auto_ptr< _Up > &&__u) noexcept`
- `unique_ptr (const unique_ptr &)=delete`
- `~unique_ptr () noexcept`
- `pointer get () const noexcept`
- `deleter_type & get_deleter () noexcept`

- `const deleter_type & get_deleter () const noexcept`
- `operator bool () const noexcept`
- `add_lvalue_reference< element_type >::type operator* () const`
- `pointer operator-> () const noexcept`
- `unique_ptr & operator= (unique_ptr &&__u) noexcept`
- `template<typename _Up, typename _Ep >  
enable_if<__and< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && > >::value,  
unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept`
- `unique_ptr & operator= (nullptr_t) noexcept`
- `unique_ptr & operator= (const unique_ptr &)=delete`
- `pointer release () noexcept`
- `void reset (pointer __p=pointer()) noexcept`
- `void swap (unique_ptr &__u) noexcept`

### 5.1082.1 Detailed Description

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
class std::unique_ptr< _Tp, _Dp >
```

20.7.1.2 `unique_ptr` for single objects.

Definition at line 168 of file `unique_ptr.h`.

### 5.1082.2 Constructor & Destructor Documentation

#### 5.1082.2.1 `unique_ptr()` [1/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>
constexpr std::unique_ptr< _Tp, _Dp >::unique_ptr ( ) [inline], [noexcept]
```

Default constructor, creates a `unique_ptr` that owns nothing.


Definition at line 194 of file `unique_ptr.h`.

#### 5.1082.2.2 `unique_ptr()` [2/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p ) [inline], [explicit], [noexcept]
```

Takes ownership of a pointer.

## Parameters

<a href="#"></a> <code>__p</code>	A pointer to an object of <code>element_type</code>
---	---

The deleter will be value-initialized.



Definition at line 207 of file `unique_ptr.h`.

## 5.1082.2.3 unique\_ptr() [3/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p,
    typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter↵
_type & >::type __d ) [inline], [noexcept]
```

Takes ownership of a pointer.

## Parameters

<a href="#"></a> <code>__p</code>	A pointer to an object of <code>element_type</code>
<a href="#"></a> <code>__d</code>	A reference to a deleter.

The deleter will be initialized with `__d`



Definition at line 218 of file `unique_ptr.h`.

## 5.1082.2.4 unique\_ptr() [4/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
    pointer __p,
    typename remove_reference< deleter_type >::type && __d ) [inline], [noexcept]
```

Takes ownership of a pointer.

## Parameters

<a href="#"></a> <code>__p</code>	A pointer to an object of <code>element_type</code>
<a href="#"></a> <code>__d</code>	An rvalue reference to a deleter.

The deleter will be initialized with `std::move(__d)`

Definition at line 230 of file `unique_ptr.h`.

#### 5.1082.2.5 `unique_ptr()` [5/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>
constexpr std::unique_ptr<_Tp, _Dp>::unique_ptr (
    nullptr_t ) [inline], [noexcept]
```

Creates a `unique_ptr` that owns nothing.

Definition at line 239 of file `unique_ptr.h`.

#### 5.1082.2.6 `unique_ptr()` [6/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr<_Tp, _Dp>::unique_ptr (
    unique_ptr<_Tp, _Dp> && __u ) [inline], [noexcept]
```

Move constructor.

Definition at line 244 of file `unique_ptr.h`.

#### 5.1082.2.7 `unique_ptr()` [7/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep, typename = _Require<__safe_conversion_up<_Up, _Ep>,
typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>><
::type>>>
std::unique_ptr<_Tp, _Dp>::unique_ptr (
    unique_ptr<_Up, _Ep> && __u ) [inline], [noexcept]
```

Converting constructor from another type.

Requires that the pointer owned by `__u` is convertible to the type of pointer owned by this object, `__u` does not own an array, and `__u` has a compatible deleter type.

Definition at line 258 of file `unique_ptr.h`.

### 5.1082.2.8 ~unique\_ptr()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr<_Tp, _Dp>::~~unique_ptr ( ) [inline], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

Definition at line 273 of file unique\_ptr.h.

## 5.1082.3 Member Function Documentation

### 5.1082.3.1 get()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr<_Tp, _Dp>::get (
    void ) const [inline], [noexcept]
```

Return the stored pointer.

Definition at line 345 of file unique\_ptr.h.

Referenced by std::unique\_ptr<\_Result<\_Res>>::operator bool(), std::unique\_ptr<\_Tp[], \_Dp>::operator bool(), std::unique\_ptr<\_Result<\_Res>>::release(), and std::unique\_ptr<\_Tp[], \_Dp>::release().

### 5.1082.3.2 get\_deleter() [1/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
deleter_type& std::unique_ptr<_Tp, _Dp>::get_deleter ( ) [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 350 of file unique\_ptr.h.

Referenced by std::unique\_ptr<\_Result<\_Res>>::operator=(), std::unique\_ptr<\_Tp[], \_Dp>::operator=(), std::unique\_ptr<\_Result<\_Res>>::reset(), std::unique\_ptr<\_Tp[], \_Dp>::reset(), std::unique\_ptr<\_Result<\_Res>>::~~unique\_ptr(), and std::unique\_ptr<\_Tp[], \_Dp>::~~unique\_ptr().

### 5.1082.3.3 get\_deleter() [2/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
const deleter_type& std::unique_ptr<_Tp, _Dp>::get_deleter ( ) const [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 355 of file unique\_ptr.h.

#### 5.1082.3.4 operator bool()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr<_Tp, _Dp>::operator bool ( ) const [inline], [explicit], [noexcept]
```

Return true if the stored pointer is not null.

Definition at line 359 of file unique\_ptr.h.

#### 5.1082.3.5 operator\*()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
add_lvalue_reference<element_type>::type std::unique_ptr<_Tp, _Dp>::operator* ( ) const [inline]
```

Dereference the stored pointer.

Definition at line 329 of file unique\_ptr.h.

#### 5.1082.3.6 operator->()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr<_Tp, _Dp>::operator-> ( ) const [inline], [noexcept]
```

Return the stored pointer.


Definition at line 337 of file unique\_ptr.h.

#### 5.1082.3.7 operator=() [1/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr& std::unique_ptr<_Tp, _Dp>::operator= (
    unique_ptr<_Tp, _Dp> && __u ) [inline], [noexcept]
```

Move assignment operator.

##### Parameters

 <i>__u</i>	The object to transfer ownership from.
---	--

Invokes the deleter first if this object owns a pointer.

Definition at line 290 of file unique\_ptr.h.

## 5.1082.3.8 operator=() [2/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep >
enable_if< __and< __safe_conversion_up<_Up, _Ep>, is_assignable<deleter_type&, _Ep&&> >↵
::value, unique_ptr&>::type std::unique_ptr<_Tp, _Dp>::operator= (
    unique_ptr<_Up, _Ep> && __u ) [inline], [noexcept]
```

Assignment from another type.

## Parameters

<u>↵</u> <u>u</u>	The object to transfer ownership from, which owns a convertible pointer to a non-array object.
----------------------	--

Invokes the deleter first if this object owns a pointer.

Definition at line 310 of file unique\_ptr.h.

## 5.1082.3.9 operator=() [3/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr& std::unique_ptr<_Tp, _Dp>::operator= (
    nullptr_t ) [inline], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

Definition at line 319 of file unique\_ptr.h.

## 5.1082.3.10 release()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr<_Tp, _Dp>::release ( ) [inline], [noexcept]
```

Release ownership of any stored pointer.

Definition at line 366 of file unique\_ptr.h.

## 5.1082.3.11 reset()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr<_Tp, _Dp>::reset (
    pointer __p = pointer() ) [inline], [noexcept]
```

Replace the stored pointer.



## Parameters

<code>_p</code>	The new pointer to store.
-----------------	---------------------------

The deleter will be invoked if a pointer is already owned.

Definition at line 380 of file `unique_ptr.h`.

Referenced by `std::unique_ptr<_Result<_Res>>::operator=()`, and `std::unique_ptr<_Tp[],_Dp>::operator=()`.

5.1082.3.12 `swap()`

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr<_Tp, _Dp>::swap (
    unique_ptr<_Tp, _Dp> & __u ) [inline], [noexcept]
```

Exchange the pointer and deleter with another object.

Definition at line 390 of file `unique_ptr.h`.

Referenced by `std::unique_ptr<_Result<_Res>>::reset()`, and `std::unique_ptr<_Tp[],_Dp>::reset()`.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

5.1083 `std::unique_ptr<_Tp[],_Dp>` Class Template Reference

## Public Types

- `template<typename _Up>`  
`using __safe_conversion_raw = __and_<__or_<__or_<is_same<_Up, pointer>, is_same<_Up, nullptr_t>>, __and_<is_pointer<_Up>, is_same<pointer, element_type*>, is_convertible<typename remove_pointer<_Up>::type(*)[], element_type(*)[]>>>>`
- `template<typename _Up, typename _Ep, typename _UPtr = unique_ptr<_Up, _Ep>, typename _UP_pointer = typename _UPtr::pointer, typename _UP_element_type = typename _UPtr::element_type>`  
`using __safe_conversion_up = __and_<is_array<_Up>, is_same<pointer, element_type*>, is_same<_UP_pointer, _UP_element_type*>, is_convertible<_UP_element_type(*)[], element_type(*)[]>>>`
- using `deleter_type` = `_Dp`
- using `element_type` = `_Tp`
- using `pointer` = `typename __uniq_ptr_impl<_Tp, _Dp>::pointer`

## Public Member Functions

- template<typename \_Up = \_Dp, typename = \_DeleterConstraint<\_Up>>  
constexpr [unique\\_ptr](#) () noexcept
- template<typename \_Up, typename \_Vp = \_Dp, typename = \_DeleterConstraint<\_Vp>, typename = typename enable\_if< \_\_safe\_conversion\_raw<\_Up>::value, bool>::type>  
[unique\\_ptr](#) (\_Up \_\_p) noexcept
- template<typename \_Up, typename = typename enable\_if< \_\_safe\_conversion\_raw<\_Up>::value, bool>::type>  
[unique\\_ptr](#) (\_Up \_\_p, typename [conditional](#)< [is\\_reference](#)< deleter\_type >::value, deleter\_type, const deleter\_type & >::type \_\_d) noexcept
- template<typename \_Up, typename = typename enable\_if< \_\_safe\_conversion\_raw<\_Up>::value, bool>::type>  
[unique\\_ptr](#) (\_Up \_\_p, typename [remove\\_reference](#)< deleter\_type >::type &&\_\_d) noexcept
- [unique\\_ptr](#) ([unique\\_ptr](#) &&\_\_u) noexcept
- template<typename \_Up = \_Dp, typename = \_DeleterConstraint<\_Up>>  
constexpr [unique\\_ptr](#) (nullptr\_t) noexcept
- template<typename \_Up, typename \_Ep, typename = \_Require< \_\_safe\_conversion\_up<\_Up, \_Ep>, typename [conditional](#)<[is\\_reference](#)<\_Dp>::value, [is\\_same](#)<\_Ep, \_Dp>, [is\\_convertible](#)<\_Ep, \_Dp>>::type>>  
[unique\\_ptr](#) ([unique\\_ptr](#)< \_Up, \_Ep > &&\_\_u) noexcept
- [unique\\_ptr](#) (const [unique\\_ptr](#) &)=delete
- ~[unique\\_ptr](#) ()
- pointer [get](#) () const noexcept
- deleter\_type & [get\\_deleter](#) () noexcept
- const deleter\_type & [get\\_deleter](#) () const noexcept
- [operator bool](#) () const noexcept
- [unique\\_ptr](#) & [operator=](#) ([unique\\_ptr](#) &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep >  
[enable\\_if](#)< \_\_and< \_\_safe\_conversion\_up< \_Up, \_Ep >, [is\\_assignable](#)< deleter\_type &, \_Ep && > >::value,  
[unique\\_ptr](#) & >::type [operator=](#) ([unique\\_ptr](#)< \_Up, \_Ep > &&\_\_u) noexcept
- [unique\\_ptr](#) & [operator=](#) (nullptr\_t) noexcept
- [unique\\_ptr](#) & [operator=](#) (const [unique\\_ptr](#) &)=delete
- [std::add\\_lvalue\\_reference](#)< element\_type >::type [operator\[\]](#) (size\_t \_\_i) const
- pointer [release](#) () noexcept
- template<typename \_Up, typename = \_Require< \_\_or<[is\\_same](#)<\_Up, pointer>, \_\_and<[is\\_same](#)<pointer, element\_type\*>, [is\\_convertible](#)< typename [remove\\_pointer](#)<\_Up>::type(\*)[], element\_type(\*)[] > > >>  
void [reset](#) (\_Up \_\_p) noexcept
- void [reset](#) (nullptr\_t=nullptr) noexcept
- void [swap](#) ([unique\\_ptr](#) &\_\_u) noexcept

## 5.1083.1 Detailed Description

```
template<typename _Tp, typename _Dp>
class std::unique_ptr< _Tp[], _Dp >
```

20.7.1.3 [unique\\_ptr](#) for array objects with a runtime length

Definition at line 406 of file [unique\\_ptr.h](#).

## 5.1083.2 Constructor &amp; Destructor Documentation

**5.1083.2.1** `unique_ptr()` [1/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>
constexpr std::unique_ptr< _Tp[], _Dp >::unique_ptr ( ) [inline], [noexcept]
```

Default constructor, creates a `unique_ptr` that owns nothing.

Definition at line 460 of file `unique_ptr.h`.

**5.1083.2.2** `unique_ptr()` [2/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename
enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    _Up __p ) [inline], [explicit], [noexcept]
```

Takes ownership of a pointer.

**Parameters**

<code>__p</code>	A pointer to an array of a type safely convertible to an array of <code>element_type</code>
------------------	---

The deleter will be value-initialized.

Definition at line 477 of file `unique_ptr.h`.

**5.1083.2.3** `unique_ptr()` [3/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename = typename enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    _Up __p,
    typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter<
_type & >::type __d ) [inline], [noexcept]
```

Takes ownership of a pointer.

**Parameters**

<code>__p</code>	A pointer to an array of a type safely convertible to an array of <code>element_type</code>
<code>__d</code>	A reference to a deleter.

The deleter will be initialized with `__d`

Definition at line 492 of file `unique_ptr.h`.

#### 5.1083.2.4 unique\_ptr() [4/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename = typename enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    _Up __p,
    typename remove_reference< deleter_type >::type && __d ) [inline], [noexcept]
```

Takes ownership of a pointer.

##### Parameters

<code>__p</code>	A pointer to an array of a type safely convertible to an array of <code>element_type</code>
<code>__d</code>	A reference to a deleter.

The deleter will be initialized with `std::move(__d)`

Definition at line 508 of file `unique_ptr.h`.

#### 5.1083.2.5 unique\_ptr() [5/6]

```
template<typename _Tp , typename _Dp >
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    unique_ptr< _Tp[], _Dp > && __u ) [inline], [noexcept]
```

Move constructor.

Definition at line 515 of file `unique_ptr.h`.

#### 5.1083.2.6 unique\_ptr() [6/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up = _Dp, typename = _DeleterConstraint<_Up>>
constexpr std::unique_ptr< _Tp[], _Dp >::unique_ptr (
    nullptr_t ) [inline], [noexcept]
```

Creates a `unique_ptr` that owns nothing.

Definition at line 521 of file `unique_ptr.h`.

### 5.1083.2.7 ~unique\_ptr()

```
template<typename _Tp , typename _Dp >
std::unique_ptr< _Tp[], _Dp >::~~unique_ptr ( ) [inline]
```

Destructor, invokes the deleter if the stored pointer is not null.

Definition at line 533 of file unique\_ptr.h.

References `std::unique_ptr< _Tp, _Dp >::get_deleter()`.

## 5.1083.3 Member Function Documentation

### 5.1083.3.1 get()

```
template<typename _Tp , typename _Dp >
pointer std::unique_ptr< _Tp[], _Dp >::get (
    void ) const [inline], [noexcept]
```

Return the stored pointer.

Definition at line 597 of file unique\_ptr.h.

### 5.1083.3.2 get\_deleter() [1/2]

```
template<typename _Tp , typename _Dp >
deleter_type& std::unique_ptr< _Tp[], _Dp >::get_deleter ( ) [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 602 of file unique\_ptr.h.

### 5.1083.3.3 get\_deleter() [2/2]

```
template<typename _Tp , typename _Dp >
const deleter_type& std::unique_ptr< _Tp[], _Dp >::get_deleter ( ) const [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 607 of file unique\_ptr.h.

#### 5.1083.3.4 operator bool()

```
template<typename _Tp , typename _Dp >
std::unique_ptr< _Tp[], _Dp >::operator bool ( ) const [inline], [explicit], [noexcept]
```

Return `true` if the stored pointer is not null.

Definition at line 611 of file `unique_ptr.h`.

References `std::unique_ptr< _Tp, _Dp >::get()`.

#### 5.1083.3.5 operator=() [1/3]

```
template<typename _Tp , typename _Dp >
unique_ptr& std::unique_ptr< _Tp[], _Dp >::operator= (
    unique_ptr< _Tp[], _Dp > && __u ) [inline], [noexcept]
```

Move assignment operator.

##### Parameters

<code>__u</code>	The object to transfer ownership from.
------------------	--

Invokes the deleter first if this object owns a pointer.

Definition at line 550 of file `unique_ptr.h`.

References `std::unique_ptr< _Tp, _Dp >::get_deleter()`, and `std::unique_ptr< _Tp, _Dp >::reset()`.

#### 5.1083.3.6 operator=() [2/3]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename _Ep >
enable_if<__and<__safe_conversion_up<_Up, _Ep>, is_assignable<deleter_type&, _Ep&&> >::value,
unique_ptr&>::type std::unique_ptr< _Tp[], _Dp >::operator= (
    unique_ptr< _Up, _Ep > && __u ) [inline], [noexcept]
```

Assignment from another type.

##### Parameters

<code>__u</code>	The object to transfer ownership from, which owns a convertible pointer to an array object.
------------------	---

Invokes the deleter first if this object owns a pointer.

Definition at line 570 of file `unique_ptr.h`.

References `std::unique_ptr<_Tp, _Dp>::get_deleter()`, and `std::unique_ptr<_Tp, _Dp>::reset()`.

#### 5.1083.3.7 `operator=()` [3/3]

```
template<typename _Tp , typename _Dp >
unique_ptr& std::unique_ptr<_Tp[], _Dp>::operator= (
    nullptr_t ) [inline], [noexcept]
```

Reset the `unique_ptr` to empty, invoking the deleter if necessary.

Definition at line 579 of file `unique_ptr.h`.

References `std::unique_ptr<_Tp, _Dp>::reset()`.

#### 5.1083.3.8 `operator[]()`

```
template<typename _Tp , typename _Dp >
std::add_lvalue_reference<element_type>::type std::unique_ptr<_Tp[], _Dp>::operator[] (
    size_t __i ) const [inline]
```

Access an element of owned array.

Definition at line 589 of file `unique_ptr.h`.

#### 5.1083.3.9 `release()`

```
template<typename _Tp , typename _Dp >
pointer std::unique_ptr<_Tp[], _Dp>::release ( ) [inline], [noexcept]
```

Release ownership of any stored pointer.

Definition at line 618 of file `unique_ptr.h`.

References `std::unique_ptr<_Tp, _Dp>::get()`.

#### 5.1083.3.10 `reset()`

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename = _Require<__or_<is_same<_Up, pointer>, __and_<is_same<pointer,
element_type*>, is_pointer<_Up>, is_convertible<typename remove_pointer<_Up>::type(*)[], element←
_type(*)[]>>>>>
void std::unique_ptr<_Tp[], _Dp>::reset (
    _Up __p ) [inline], [noexcept]
```

Replace the stored pointer.

## Parameters

<a href="#"><code>_p</code></a>	The new pointer to store.
---------------------------------	---------------------------

The deleter will be invoked if a pointer is already owned.

Definition at line 644 of file `unique_ptr.h`.

References `std::unique_ptr< _Tp, _Dp >::get_deleter()`, and `std::unique_ptr< _Tp, _Dp >::swap()`.

## 5.1083.3.11 swap()

```
template<typename _Tp , typename _Dp >
void std::unique_ptr< _Tp[], _Dp >::swap (
    unique_ptr< _Tp[], _Dp > & __u ) [inline], [noexcept]
```

Exchange the pointer and deleter with another object.

Definition at line 660 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 5.1084 std::unordered\_map&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::mapped\_type [mapped\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)



## Public Member Functions

- `unordered_map` ()=default
- `unordered_map` (`size_type` \_\_n, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_map` (const `unordered_map` &)=default
- `unordered_map` (`unordered_map` &&)=default
- `unordered_map` (const `allocator_type` &\_\_a)
- `unordered_map` (const `unordered_map` &\_\_umap, const `allocator_type` &\_\_a)
- `unordered_map` (`unordered_map` &&\_\_umap, const `allocator_type` &\_\_a)
- `unordered_map` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_map` (`size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_map` (`size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `unordered_map` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_map` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `iterator begin` () noexcept
- `local_iterator begin` (`size_type` \_\_n)
- `size_type bucket` (const `key_type` &\_\_key) const
- `size_type bucket_count` () const noexcept
- `size_type bucket_size` (`size_type` \_\_n) const
- `void clear` () noexcept
- `size_type count` (const `key_type` &\_\_x) const
- `template<typename... _Args>`  
`std::pair`< `iterator`, bool > `emplace` (\_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator emplace_hint` (const `iterator` \_\_pos, \_Args &&... \_\_args)
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `local_iterator end` (`size_type` \_\_n)
- `size_type erase` (const `key_type` &\_\_x)
- `iterator erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `allocator_type get_allocator` () const noexcept
- `hasher hash_function` () const
- `template<typename _InputIterator >`  
`void insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `void insert` (`initializer_list`< `value_type` > \_\_l)
- `key_equal key_eq` () const
- `float load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- `float max_load_factor` () const noexcept
- `void max_load_factor` (float \_\_z)



- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const

- `mapped_type & operator[]` (const `key_type` &\_\_k)
- `mapped_type & operator[]` (`key_type` &&\_\_k)

- `mapped_type & at` (const `key_type` &\_\_k)
- `const mapped_type & at` (const `key_type` &\_\_k) const

- `const_local_iterator begin` (size\_type \_\_n) const
- `const_local_iterator cbegin` (size\_type \_\_n) const

- `const_local_iterator end` (size\_type \_\_n) const
- `const_local_iterator cend` (size\_type \_\_n) const

#### Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 >`  
`bool operator==` (const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

#### 5.1084.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc =  
allocator<std::pair<const _Key, _Tp>>>>  
class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

## Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

Definition at line 102 of file `unordered_map.h`.

## 5.1084.2 Member Typedef Documentation

## 5.1084.2.1 allocator\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >::allocator\_type
```

Public typedefs.

Definition at line 116 of file `unordered_map.h`.

## 5.1084.2.2 const\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >::const\_iterator
```

Iterator-related typedefs.

Definition at line 126 of file `unordered_map.h`.

### 5.1084.2.3 const\_local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

Definition at line 128 of file unordered\_map.h.

### 5.1084.2.4 const\_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_pointer std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 122 of file unordered\_map.h.

### 5.1084.2.5 const\_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference
```

Iterator-related typedefs.

Definition at line 124 of file unordered\_map.h.

### 5.1084.2.6 difference\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type
```

Iterator-related typedefs.

Definition at line 130 of file unordered\_map.h.

#### 5.1084.2.7 hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 114 of file unordered\_map.h.

#### 5.1084.2.8 iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 125 of file unordered\_map.h.

#### 5.1084.2.9 key\_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 115 of file unordered\_map.h.

#### 5.1084.2.10 key\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::key_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 111 of file unordered\_map.h.

#### 5.1084.2.11 local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator
```

Iterator-related typedefs.

Definition at line 127 of file unordered\_map.h.

#### 5.1084.2.12 mapped\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type
```

Public typedefs.

Definition at line 113 of file unordered\_map.h.

#### 5.1084.2.13 pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 121 of file unordered\_map.h.

#### 5.1084.2.14 reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 123 of file unordered\_map.h.

## 5.1084.2.15 size\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 129 of file unordered\_map.h.

## 5.1084.2.16 value\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::value_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 112 of file unordered\_map.h.

## 5.1084.3 Constructor &amp; Destructor Documentation

## 5.1084.3.1 unordered\_map() [1/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map ( ) [default]
```

Default constructor.

## 5.1084.3.2 unordered\_map() [2/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Default constructor creates no elements.



## Parameters

<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 151 of file `unordered_map.h`.

5.1084.3.3 `unordered_map()` [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator >
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an `unordered_map` from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_map` consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first,__last)`).

Definition at line 172 of file `unordered_map.h`.

5.1084.3.4 `unordered_map()` [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

## 5.1084.3.5 unordered\_map() [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

## 5.1084.3.6 unordered\_map() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    const allocator_type & __a ) [inline], [explicit]
```

Creates an unordered\_map with no elements.

## Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 191 of file unordered\_map.h.

## 5.1084.3.7 unordered\_map() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_map from an initializer\_list.

## Parameters

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq1</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_map` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `__l.size()`).

Definition at line 226 of file `unordered_map.h`.

#### 5.1084.4 Member Function Documentation

##### 5.1084.4.1 `at()` [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::at (
    const key_type & __k ) [inline]
```

Access to `unordered_map` data.

##### Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

##### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

##### Exceptions

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 990 of file `unordered_map.h`.

##### 5.1084.4.2 `at()` [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::at (
    const key_type & __k ) const [inline]
```

Access to `unordered_map` data.

##### Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

**Returns**

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

**Exceptions**

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 994 of file `unordered_map.h`.

**5.1084.4.3 begin()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_map`.

Definition at line 324 of file `unordered_map.h`.

**5.1084.4.4 begin()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

Definition at line 333 of file `unordered_map.h`.

**5.1084.4.5 begin()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read/write local iterator.

Definition at line 1035 of file unordered\_map.h.

**5.1084.4.6 begin()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

<code>_↵</code>	The bucket index.
<code>_n</code>	

**Returns**

A read-only local iterator.

Definition at line 1046 of file unordered\_map.h.

**5.1084.4.7 bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_map.

Definition at line 1002 of file unordered\_map.h.

**5.1084.4.8 cbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_map.

Definition at line 337 of file unordered\_map.h.

## 5.1084.4.9 cbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

<code>_↵</code>	The bucket index.
<code>_n</code>	

## Returns

A read-only local iterator.

Definition at line 1050 of file unordered\_map.h.

## 5.1084.4.10 cend() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

Definition at line 359 of file unordered\_map.h.

## 5.1084.4.11 cend() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

<code>_↵</code>	The bucket index.
<code>_n</code>	

**Returns**

A read-only local iterator.

Definition at line 1076 of file unordered\_map.h.

**5.1084.4.12 clear()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in an unordered\_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 842 of file unordered\_map.h.

**5.1084.4.13 count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements.

**Parameters**

<code>__x</code>	Key to count.
------------------	---------------

**Returns**

Number of elements with specified key.

This function only makes sense for unordered\_multimap; for unordered\_map the result will either be 0 (not present) or 1 (present).

Definition at line 938 of file unordered\_map.h.

5.1084.4.14 `emplace()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

## Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 387 of file `unordered_map.h`.

5.1084.4.15 `emplace_hint()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).



This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 418 of file `unordered_map.h`.

#### 5.1084.4.16 `empty()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::empty ( ) const [inline], [noexcept]
```

Returns true if the `unordered_map` is empty.

Definition at line 304 of file `unordered_map.h`.

#### 5.1084.4.17 `end()` [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_map`.

Definition at line 346 of file `unordered_map.h`.

#### 5.1084.4.18 `end()` [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_map`.

Definition at line 355 of file `unordered_map.h`.

#### 5.1084.4.19 `end()` [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end (
    size_type __n ) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

## Parameters

<code>_↵</code>	The bucket index.
<code>_n</code>	

## Returns

A read/write local iterator.

Definition at line 1061 of file unordered\_map.h.

## 5.1084.4.20 end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

<code>_↵</code>	The bucket index.
<code>_n</code>	

## Returns

A read-only local iterator.

Definition at line 1072 of file unordered\_map.h.

## 5.1084.4.21 equal\_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, iterator> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range
(
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>_↵</code>	Key to be located.
<code>_X</code>	

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

Definition at line 951 of file `unordered_map.h`.

**5.1084.4.22 `equal_range()`** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::pair<const_iterator, const_iterator> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::equal_range (
    const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>_↵</code>	Key to be located.
<code>_X</code>	

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

Definition at line 955 of file `unordered_map.h`.

**5.1084.4.23 `erase()`** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position ) [inline]
```

Erases an element from an `unordered_map`.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_map`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 792 of file `unordered_map.h`.

5.1084.4.24 `erase()` [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    iterator __position ) [inline]
```

Erases an element from an `unordered_map`.

## Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_map`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 797 of file `unordered_map.h`.

5.1084.4.25 `erase()` [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

**Parameters**

<code>_↵</code>	Key of element to be erased.
<code>_X</code>	

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_map`. For an `unordered_map` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 814 of file `unordered_map.h`.

**5.1084.4.26 erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_map`.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_map`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 832 of file `unordered_map.h`.

**5.1084.4.27** find() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in an unordered\_map.

**Parameters**

<code>_↵</code>	Key to be located.
<code>_X</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 920 of file `unordered_map.h`.

**5.1084.4.28 find()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_map`.

**Parameters**

<code>_↵</code>	Key to be located.
<code>_X</code>	

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 924 of file `unordered_map.h`.

**5.1084.4.29 get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator ( ) const
[inline], [noexcept]
```

Returns the allocator object used by the `unordered_map`.

Definition at line 297 of file `unordered_map.h`.

## 5.1084.4.30 hash\_function()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]
```

Returns the hash functor object with which the unordered\_map was constructed.

Definition at line 896 of file unordered\_map.h.

## 5.1084.4.31 insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).
------------------	--

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

Definition at line 579 of file unordered\_map.h.

## 5.1084.4.32 insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    value_type && __x ) [inline]
```

Attempts to insert a std::pair into the unordered\_map.



**Parameters**

<code>_↵</code> <code>_X</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------------------------	---

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 585 of file `unordered_map.h`.

**5.1084.4.33 insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, pair<iterator, bool> > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _Pair && __x ) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

<code>_↵</code> <code>_X</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------------------------	---

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 591 of file `unordered_map.h`.

## 5.1084.4.34 insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    const value_type & __x ) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 618 of file unordered\_map.h.

## 5.1084.4.35 insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x ) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 624 of file `unordered_map.h`.

**5.1084.4.36 insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::unordered_map< _Key,
_Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    _Pair && __x ) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 629 of file `unordered_map.h`.

**5.1084.4.37 insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator >
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 644 of file unordered\_map.h.

**5.1084.4.38 insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the unordered\_map.

**Parameters**

<code>↵</code> <code>↵</code> <code>↵</code> <code>↵</code> <code>/</code>	A std::initializer_list<value_type> of elements to be inserted.
--	---

Complexity similar to that of the range constructor.

Definition at line 655 of file unordered\_map.h.

**5.1084.4.39 key\_eq()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]
```

Returns the key comparison object with which the `unordered_map` was constructed.

Definition at line 902 of file `unordered_map.h`.

#### 5.1084.4.40 `load_factor()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1084 of file `unordered_map.h`.

#### 5.1084.4.41 `max_bucket_count()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline],
[noexcept]
```

Returns the maximum number of buckets of the `unordered_map`.

Definition at line 1007 of file `unordered_map.h`.

#### 5.1084.4.42 `max_load_factor()` [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline],
[noexcept]
```

Returns a positive number that the `unordered_map` tries to keep the load factor less than or equal to.

Definition at line 1090 of file `unordered_map.h`.

#### 5.1084.4.43 `max_load_factor()` [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z ) [inline]
```

Change the `unordered_map` maximum load factor.

## Parameters

<code>_Z</code>	The new maximum load factor.
-----------------	------------------------------

Definition at line 1098 of file unordered\_map.h.

## 5.1084.4.44 max\_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size ( ) const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_map.

Definition at line 314 of file unordered\_map.h.

## 5.1084.4.45 operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

## 5.1084.4.46 operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

## 5.1084.4.47 operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > & __l ) [inline]
```

Unordered\_map list assignment operator.

## Parameters

$\leftarrow$	An initializer_list.
$\_ \leftarrow$	
$\leftarrow$	
$\_ \leftarrow$	
$/$	

This function fills an unordered\_map with copies of the elements in the initializer list  $\_ /$ .

Note that the assignment completely changes the unordered\_map and that the resulting unordered\_map's size is the same as the number of elements assigned.

Definition at line 289 of file unordered\_map.h.

## 5.1084.4.48 operator[]() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
    const key_type & __k ) [inline]
```

Subscript (  $[]$  ) access to unordered\_map data.

## Parameters

$\_ \leftarrow$ $\_ k$	The key for which data should be retrieved.
---------------------------	---

## Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript (  $[]$  ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 973 of file unordered\_map.h.

## 5.1084.4.49 operator[]() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
    key_type && __k ) [inline]
```

Subscript (  $[]$  ) access to unordered\_map data.

## Parameters

<code>↵ _k</code>	The key for which data should be retrieved.
-----------------------	---

## Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] )operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 977 of file unordered\_map.h.

## 5.1084.4.50 rehash()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
    size_type __n ) [inline]
```

May rehash the unordered\_map.

## Parameters

<code>↵ _n</code>	The new number of buckets.
-----------------------	----------------------------

Rehash will occur only if the new number of buckets respect the unordered\_map maximum load factor.

Definition at line 1109 of file unordered\_map.h.

## 5.1084.4.51 reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
    size_type __n ) [inline]
```

Prepare the unordered\_map for a specified number of elements.



**Parameters**

<code>_↵</code> <code>_n</code>	Number of elements required.
------------------------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1120 of file `unordered_map.h`.

**5.1084.4.52 `size()`**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size ( ) const    [inline], [noexcept]
```

Returns the size of the `unordered_map`.

Definition at line 309 of file `unordered_map.h`.

**5.1084.4.53 `swap()`**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
    unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x )    [inline], [noexcept]
```

Swaps data with another `unordered_map`.

**Parameters**

<code>_↵</code> <code>_X</code>	An <code>unordered_map</code> of the same element and allocator types.
------------------------------------	--

This exchanges the elements between two `unordered_map` in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 856 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 5.1085 std::unordered\_multimap&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
  - typedef \_Hashtable::value\_type [value\\_type](#)
  - typedef \_Hashtable::mapped\_type [mapped\\_type](#)
  - typedef \_Hashtable::hasher [hasher](#)
  - typedef \_Hashtable::key\_equal [key\\_equal](#)
  - typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- 
- typedef \_Hashtable::pointer [pointer](#)
  - typedef \_Hashtable::const\_pointer [const\\_pointer](#)
  - typedef \_Hashtable::reference [reference](#)
  - typedef \_Hashtable::const\_reference [const\\_reference](#)
  - typedef \_Hashtable::iterator [iterator](#)
  - typedef \_Hashtable::const\_iterator [const\\_iterator](#)
  - typedef \_Hashtable::local\_iterator [local\\_iterator](#)
  - typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
  - typedef \_Hashtable::size\_type [size\\_type](#)
  - typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_multimap](#) ()=default
- [unordered\\_multimap](#) (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[unordered\\_multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [unordered\\_multimap](#) (const unordered\_multimap &)=default
- [unordered\\_multimap](#) (unordered\_multimap &&)=default
- [unordered\\_multimap](#) (const allocator\_type &\_\_a)
- **unordered\_multimap** (const unordered\_multimap &\_\_ummap, const allocator\_type &\_\_a)
- **unordered\_multimap** (unordered\_multimap &&\_\_ummap, const allocator\_type &\_\_a)
- [unordered\\_multimap](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)

- **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
  - **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
  - [iterator](#) **begin** () noexcept
  - [local\\_iterator](#) **begin** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **bucket** (const [key\\_type](#) &\_\_key) const
  - [size\\_type](#) **bucket\_count** () const noexcept
  - [size\\_type](#) **bucket\_size** ([size\\_type](#) \_\_n) const
  - void **clear** () noexcept
  - [size\\_type](#) **count** (const [key\\_type](#) &\_\_x) const
  - template<typename... \_Args>  
[iterator](#) **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - [iterator](#) **end** () noexcept
  - [local\\_iterator](#) **end** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
  - [iterator](#) **erase** (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
  - [allocator\\_type](#) **get\_allocator** () const noexcept
  - [hasher](#) **hash\_function** () const
  - template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void **insert** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [key\\_equal](#) **key\_eq** () const
  - float **load\_factor** () const noexcept
  - [size\\_type](#) **max\_bucket\_count** () const noexcept
  - float **max\_load\_factor** () const noexcept
  - void **max\_load\_factor** (float \_\_z)
  - [size\\_type](#) **max\_size** () const noexcept
  - [unordered\\_multimap](#) & **operator=** (const [unordered\\_multimap](#) &)=default
  - [unordered\\_multimap](#) & **operator=** ([unordered\\_multimap](#) &&)=default
  - [unordered\\_multimap](#) & **operator=** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - void **rehash** ([size\\_type](#) \_\_n)
  - void **reserve** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **size** () const noexcept
  - void **swap** ([unordered\\_multimap](#) &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- [const\\_iterator](#) **begin** () const noexcept
  - [const\\_iterator](#) **cbegin** () const noexcept
- 
- [const\\_iterator](#) **end** () const noexcept
  - [const\\_iterator](#) **cend** () const noexcept

- [iterator insert](#) (const [value\\_type](#) &\_\_x)
- [iterator insert](#) ([value\\_type](#) &&\_\_x)
- [template<typename \\_Pair>](#)  
[\\_\\_enable\\_if\\_t<is\\_constructible<value\\_type, \\_Pair &&>::value, iterator>](#) [insert](#) (\_Pair &&\_\_x)
- [iterator insert](#) (const\_iterator \_\_hint, const [value\\_type](#) &\_\_x)
- [iterator insert](#) (const\_iterator \_\_hint, [value\\_type](#) &&\_\_x)
- [template<typename \\_Pair>](#)  
[\\_\\_enable\\_if\\_t<is\\_constructible<value\\_type, \\_Pair &&>::value, iterator>](#) [insert](#) (const\_iterator \_\_hint, \_Pair &&\_\_x)
- [iterator erase](#) (const\_iterator \_\_position)
- [iterator erase](#) (iterator \_\_position)
- [iterator find](#) (const [key\\_type](#) &\_\_x)
- [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
- [std::pair<iterator, iterator> equal\\_range](#) (const [key\\_type](#) &\_\_x)
- [std::pair<const\\_iterator, const\\_iterator> equal\\_range](#) (const [key\\_type](#) &\_\_x) const
- [const\\_local\\_iterator begin](#) (size\_type \_\_n) const
- [const\\_local\\_iterator cbegin](#) (size\_type \_\_n) const
- [const\\_local\\_iterator end](#) (size\_type \_\_n) const
- [const\\_local\\_iterator cend](#) (size\_type \_\_n) const

#### Friends

- [template<typename \\_Key1, typename \\_Tp1, typename \\_Hash1, typename \\_Pred1, typename \\_Alloc1>](#)  
[bool operator==](#) (const [unordered\\_multimap<\\_Key1, \\_Tp1, \\_Hash1, \\_Pred1, \\_Alloc1>](#) &, const [unordered\\_multimap<\\_Key1, \\_Tp1, \\_Hash1, \\_Pred1, \\_Alloc1>](#) &)

#### 5.1085.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc =
allocator<std::pair<const _Key, _Tp>>>>
class std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

### Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> .

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

Definition at line 73 of file `unordered_map.h`.

## 5.1085.2 Member Typedef Documentation

### 5.1085.2.1 `allocator_type`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵
::allocator_type
```

Public typedefs.

Definition at line 1247 of file `unordered_map.h`.

### 5.1085.2.2 `const_iterator`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵
::const_iterator
```

Iterator-related typedefs.

Definition at line 1257 of file `unordered_map.h`.

### 5.1085.2.3 const\_local\_iterator

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc  
>::const_local_iterator
```

Iterator-related typedefs.

Definition at line 1259 of file unordered\_map.h.

### 5.1085.2.4 const\_pointer

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::const_pointer std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >↵  
::const_pointer
```

Iterator-related typedefs.

Definition at line 1253 of file unordered\_map.h.

### 5.1085.2.5 const\_reference

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::const_reference std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >↵  
::const_reference
```

Iterator-related typedefs.

Definition at line 1255 of file unordered\_map.h.

### 5.1085.2.6 difference\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::difference_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >↵  
::difference_type
```

Iterator-related typedefs.

Definition at line 1261 of file unordered\_map.h.

#### 5.1085.2.7 hasher

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 1245 of file unordered\_map.h.

#### 5.1085.2.8 iterator

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 1256 of file unordered\_map.h.

#### 5.1085.2.9 key\_equal

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 1246 of file unordered\_map.h.

#### 5.1085.2.10 key\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 1242 of file unordered\_map.h.

### 5.1085.2.11 local\_iterator

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵  
::local_iterator
```

Iterator-related typedefs.

Definition at line 1258 of file unordered\_map.h.

### 5.1085.2.12 mapped\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::mapped_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵  
::mapped_type
```

Public typedefs.

Definition at line 1244 of file unordered\_map.h.

### 5.1085.2.13 pointer

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 1252 of file unordered\_map.h.

### 5.1085.2.14 reference

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 1254 of file unordered\_map.h.



### 5.1085.2.15 size\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 1260 of file unordered\_map.h.

### 5.1085.2.16 value\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 1243 of file unordered\_map.h.

## 5.1085.3 Constructor & Destructor Documentation

### 5.1085.3.1 unordered\_multimap() [1/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap ( ) [default]
```

Default constructor.

### 5.1085.3.2 unordered\_multimap() [2/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (   
    size_type __n,  
    const hasher & __hf = hasher(),  
    const key_equal & __eq1 = key_equal(),  
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Default constructor creates no elements.

## Parameters

<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 1281 of file `unordered_map.h`.

## 5.1085.3.3 unordered\_multimap() [3/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator >
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eqf = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an `unordered_multimap` from a range.

## Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multimap` consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first,__last)`).

Definition at line 1302 of file `unordered_map.h`.

## 5.1085.3.4 unordered\_multimap() [4/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

**5.1085.3.5 unordered\_multimap()** [5/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

**5.1085.3.6 unordered\_multimap()** [6/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    const allocator_type & __a ) [inline], [explicit]
```

Creates an unordered\_multimap with no elements.

**Parameters**

<code>↵</code>	An allocator object.
<code>_a</code>	

Definition at line 1321 of file unordered\_map.h.

**5.1085.3.7 unordered\_multimap()** [7/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_multimap from an initializer\_list.

**Parameters**

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq1</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered\_multimap consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 1356 of file unordered\_map.h.

#### 5.1085.4 Member Function Documentation

##### 5.1085.4.1 begin() [1/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered\_multimap.

Definition at line 1454 of file unordered\_map.h.

##### 5.1085.4.2 begin() [2/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 1463 of file unordered\_map.h.

##### 5.1085.4.3 begin() [3/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

#### Parameters

<code>__↔ __n</code>	The bucket index.
--------------------------	-------------------

**Returns**

A read/write local iterator.

Definition at line 1868 of file unordered\_map.h.

**5.1085.4.4 begin()** [4/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (   
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

<code>_↵ __n</code>	The bucket index.
-------------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 1879 of file unordered\_map.h.

**5.1085.4.5 bucket\_count()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count ( ) const  
[inline], [noexcept]
```

Returns the number of buckets of the unordered\_multimap.

Definition at line 1835 of file unordered\_map.h.

**5.1085.4.6 cbegin()** [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline],  
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 1467 of file unordered\_map.h.

**5.1085.4.7 cbegin()** [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (   
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

<code>_↵</code>	The bucket index.
<code>_n</code>	

**Returns**

A read-only local iterator.

Definition at line 1883 of file unordered\_map.h.

**5.1085.4.8 cend()** [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend ( ) const [inline],  
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

Definition at line 1489 of file unordered\_map.h.

**5.1085.4.9 cend()** [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (   
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

<code>_↵</code>	The bucket index.
<code>_n</code>	

**Returns**

A read-only local iterator.

Definition at line 1909 of file unordered\_map.h.

**5.1085.4.10 clear()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in an unordered\_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1718 of file unordered\_map.h.

**5.1085.4.11 count()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements.

**Parameters**

<code>_↔</code>	Key to count.
<code>_X</code>	

**Returns**

Number of elements with specified key.

Definition at line 1812 of file unordered\_map.h.

**5.1085.4.12 emplace()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.

## Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

## Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the `unordered_multimap`.

Insertion requires amortized constant time.

Definition at line 1512 of file `unordered_map.h`.

5.1085.4.13 `emplace_hint()`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_multimap`.

## Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1539 of file `unordered_map.h`.



**5.1085.4.14** `empty()`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the `unordered_multimap` is empty.

Definition at line 1434 of file `unordered_map.h`.

**5.1085.4.15** `end()` [1/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_multimap`.

Definition at line 1476 of file `unordered_map.h`.

**5.1085.4.16** `end()` [2/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end ( ) const [inline],  
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multimap`.

Definition at line 1485 of file `unordered_map.h`.

**5.1085.4.17** `end()` [3/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (   
    size_type __n ) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

**Parameters**

<code>_↵</code>	The bucket index.
<code>_n</code>	

**Returns**

A read/write local iterator.

Definition at line 1894 of file unordered\_map.h.

**5.1085.4.18 end()** [4/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

<b>↵</b> <b>_n</b>	The bucket index.
-----------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 1905 of file unordered\_map.h.

**5.1085.4.19 equal\_range()** [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, iterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_↵
_range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

**Parameters**

<b>↵</b> <b>_x</b>	Key to be located.
-----------------------	--------------------

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1823 of file unordered\_map.h.

#### 5.1085.4.20 equal\_range() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
std::pair<const_iterator, const_iterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _↵  
Alloc >::equal_range (   
    const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

##### Parameters

<code>_↵</code>	Key to be located.
<code>__x</code>	

##### Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1827 of file unordered\_map.h.

#### 5.1085.4.21 erase() [1/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (   
    const_iterator __position ) [inline]
```

Erases an element from an unordered\_multimap.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

##### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1668 of file unordered\_map.h.

#### 5.1085.4.22 erase() [2/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (   
    iterator __position ) [inline]
```

Erases an element from an unordered\_multimap.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

##### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1673 of file unordered\_map.h.

#### 5.1085.4.23 erase() [3/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (   
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

##### Parameters

<code>__x</code>	Key of elements to be erased.
------------------	-------------------------------

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1689 of file `unordered_map.h`.

**5.1085.4.24** `erase()` [4/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1708 of file `unordered_map.h`.

**5.1085.4.25** `find()` [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_multimap`.

## Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1798 of file `unordered_map.h`.

## 5.1085.4.26 find() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_multimap`.

## Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1802 of file `unordered_map.h`.

## 5.1085.4.27 get\_allocator()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator ( )
const [inline], [noexcept]
```

Returns the allocator object used by the `unordered_multimap`.

Definition at line 1427 of file `unordered_map.h`.

**5.1085.4.28** `hash_function()`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]
```

Returns the hash functor object with which the `unordered_multimap` was constructed.

Definition at line 1774 of file `unordered_map.h`.

**5.1085.4.29** `insert()` [1/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (   
    const value_type & __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

<code>↵ __x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------------	---

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1553 of file `unordered_map.h`.

**5.1085.4.30** `insert()` [2/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>  
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (   
    value_type && __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

<code>↵ __x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------------	---

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1557 of file unordered\_map.h.

**5.1085.4.31 insert()** [3/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _Pair && __x ) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

<b>↵ __x</b>	Pair to be inserted (see std::make_pair for easy creation of pairs).
------------------	--

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1562 of file unordered\_map.h.

**5.1085.4.32 insert()** [4/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    const value_type & __x ) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

<b>__hint</b>	An iterator that serves as a hint as to where the pair should be inserted.
<b>__x</b>	Pair to be inserted (see std::make_pair for easy creation of pairs).



**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1587 of file `unordered_map.h`.

**5.1085.4.33 insert()** [5/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1593 of file `unordered_map.h`.

## 5.1085.4.34 insert() [6/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    _Pair && __x ) [inline]
```

Inserts a std::pair into the unordered\_multimap.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1598 of file unordered\_map.h.

## 5.1085.4.35 insert() [7/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _InputIterator >
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 1613 of file unordered\_map.h.

#### 5.1085.4.36 insert() [8/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the unordered\_multimap.

##### Parameters

↵	A std::initializer_list<value_type> of elements to be inserted.
_↵	
↵	
_↵	
/	

Complexity similar to that of the range constructor.

Definition at line 1625 of file unordered\_map.h.

#### 5.1085.4.37 key\_eq()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]
```

Returns the key comparison object with which the unordered\_multimap was constructed.

Definition at line 1780 of file unordered\_map.h.

#### 5.1085.4.38 load\_factor()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1917 of file unordered\_map.h.

**5.1085.4.39 max\_bucket\_count()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multimap.

Definition at line 1840 of file unordered\_map.h.

**5.1085.4.40 max\_load\_factor()** [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
float std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multimap tries to keep the load factor less than or equal to.

Definition at line 1923 of file unordered\_map.h.

**5.1085.4.41 max\_load\_factor()** [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z ) [inline]
```

Change the unordered\_multimap maximum load factor.

**Parameters**

<b>_↵</b> <b>_z</b>	The new maximum load factor.
------------------------	------------------------------

Definition at line 1931 of file unordered\_map.h.

**5.1085.4.42 max\_size()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_size ( ) const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multimap.

Definition at line 1444 of file unordered\_map.h.

#### 5.1085.4.43 operator=() [1/3]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

#### 5.1085.4.44 operator=() [2/3]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

#### 5.1085.4.45 operator=() [3/3]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Unordered\_multimap list assignment operator.

#### Parameters

↵	An initializer_list.
_↵	
↵	
_↵	
/	

This function fills an unordered\_multimap with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_multimap and that the resulting unordered\_multimap's size is the same as the number of elements assigned.

Definition at line 1419 of file unordered\_map.h.

#### 5.1085.4.46 rehash()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (   
    size_type __n ) [inline]
```

May rehash the unordered\_multimap.

##### Parameters

<code>↵ __n</code>	The new number of buckets.
------------------------	----------------------------

Rehash will occur only if the new number of buckets respect the unordered\_multimap maximum load factor.

Definition at line 1942 of file unordered\_map.h.

#### 5.1085.4.47 reserve()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>  
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (   
    size_type __n ) [inline]
```

Prepare the unordered\_multimap for a specified number of elements.

##### Parameters

<code>↵ __n</code>	Number of elements required.
------------------------	------------------------------

Same as rehash(ceil(n / max\_load\_factor())).

Definition at line 1953 of file unordered\_map.h.

#### 5.1085.4.48 size()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵  
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size ( ) const [inline],
[noexcept]
```

Returns the size of the unordered\_multimap.

Definition at line 1439 of file unordered\_map.h.

#### 5.1085.4.49 swap()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
    unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another unordered\_multimap.

##### Parameters

<a href="#">↵</a>	An unordered_multimap of the same element and allocator types.
<a href="#">_X</a>	

This exchanges the elements between two unordered\_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 1732 of file unordered\_map.h.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

#### 5.1086 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

##### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)

- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

### Public Member Functions

- [unordered\\_multiset](#) ()=default
- [unordered\\_multiset](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) (const [unordered\\_multiset](#) &)=default
- [unordered\\_multiset](#) ([unordered\\_multiset](#) &&)=default
- [unordered\\_multiset](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) (const [unordered\\_multiset](#) &\_\_umset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([unordered\\_multiset](#) && \_\_umset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [size\\_type](#) [bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) [bucket\\_count](#) () const noexcept
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_n) const
- [const\\_iterator](#) [cbegin](#) () const noexcept
- [const\\_iterator](#) [cend](#) () const noexcept
- void [clear](#) () noexcept
- [size\\_type](#) [count](#) (const [key\\_type](#) &\_\_x) const
- template<typename... \_Args>  
[iterator](#) [emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) [emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [size\\_type](#) [erase](#) (const [key\\_type](#) &\_\_x)
- [iterator](#) [erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)



- `allocator_type get_allocator ()` const noexcept
- `hasher hash_function ()` const
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `key_equal key_eq ()` const
- `float load_factor ()` const noexcept
- `size_type max_bucket_count ()` const noexcept
- `float max_load_factor ()` const noexcept
- `void max_load_factor (float __z)`
- `size_type max_size ()` const noexcept
- `unordered_multiset & operator= (const unordered_multiset &)=default`
- `unordered_multiset & operator= (unordered_multiset &&)=default`
- `unordered_multiset & operator= (initializer_list< value_type > __l)`
- `void rehash (size_type __n)`
- `void reserve (size_type __n)`
- `size_type size ()` const noexcept
- `void swap (unordered_multiset &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))`

- `iterator begin ()` noexcept
- `const_iterator begin ()` const noexcept

- `iterator end ()` noexcept
- `const_iterator end ()` const noexcept

- `iterator insert (const value_type &__x)`
- `iterator insert (value_type &&__x)`

- `iterator insert (const_iterator __hint, const value_type &__x)`
- `iterator insert (const_iterator __hint, value_type &&__x)`

- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`

- [iterator find](#) (const [key\\_type](#) &\_\_x)
  - [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
- 
- [std::pair< iterator, iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x)
  - [std::pair< const\\_iterator, const\\_iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x) const
- 
- [local\\_iterator begin](#) (size\_type \_\_n)
  - [const\\_local\\_iterator begin](#) (size\_type \_\_n) const
  - [const\\_local\\_iterator cbegin](#) (size\_type \_\_n) const
- 
- [local\\_iterator end](#) (size\_type \_\_n)
  - [const\\_local\\_iterator end](#) (size\_type \_\_n) const
  - [const\\_local\\_iterator cend](#) (size\_type \_\_n) const

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 >  
bool operator== (const unordered\_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered\_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &)`

#### 5.1086.1 Detailed Description

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc =
allocator<_Value>>
class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

#### Template Parameters

<code>_Value</code>	Type of key objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__umset_hashtable`.

Definition at line 70 of file `unordered_set.h`.

## 5.1086.2 Member Typedef Documentation

### 5.1086.2.1 `allocator_type`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::allocator_type
```

Public typedefs.

Definition at line 908 of file `unordered_set.h`.

### 5.1086.2.2 `const_iterator`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::const_iterator
```

Iterator-related typedefs.

Definition at line 918 of file `unordered_set.h`.

### 5.1086.2.3 `const_local_iterator`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::const_local_iterator
```

Iterator-related typedefs.

Definition at line 920 of file `unordered_set.h`.

#### 5.1086.2.4 const\_pointer

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 914 of file unordered\_set.h.

#### 5.1086.2.5 const\_reference

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵  
::const_reference
```

Iterator-related typedefs.

Definition at line 916 of file unordered\_set.h.

#### 5.1086.2.6 difference\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵  
::difference_type
```

Iterator-related typedefs.

Definition at line 922 of file unordered\_set.h.

#### 5.1086.2.7 hasher

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 906 of file unordered\_set.h.

### 5.1086.2.8 iterator

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 917 of file `unordered_set.h`.

### 5.1086.2.9 key\_equal

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 907 of file `unordered_set.h`.

### 5.1086.2.10 key\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 904 of file `unordered_set.h`.

### 5.1086.2.11 local\_iterator

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >↵
::local_iterator
```

Iterator-related typedefs.

Definition at line 919 of file `unordered_set.h`.

#### 5.1086.2.12 pointer

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::pointer std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 913 of file unordered\_set.h.

#### 5.1086.2.13 reference

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::reference std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 915 of file unordered\_set.h.

#### 5.1086.2.14 size\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::size_type std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 921 of file unordered\_set.h.

#### 5.1086.2.15 value\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::value_type std::unordered\_multiset< _Value, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 905 of file unordered\_set.h.

### 5.1086.3 Constructor & Destructor Documentation

**5.1086.3.1 unordered\_multiset()** [1/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset ( ) [default]
```

Default constructor.

**5.1086.3.2 unordered\_multiset()** [2/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Default constructor creates no elements.

**Parameters**

<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq1</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 942 of file unordered\_set.h.

**5.1086.3.3 unordered\_multiset()** [3/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_multiset from a range.

**Parameters**

<code>__first</code>	An input iterator.
----------------------	--------------------

## Parameters

<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq1</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multiset` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 963 of file `unordered_set.h`.

## 5.1086.3.4 unordered\_multiset() [4/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

## 5.1086.3.5 unordered\_multiset() [5/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

## 5.1086.3.6 unordered\_multiset() [6/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an `unordered_multiset` from an `initializer_list`.



**Parameters**

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multiset` consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 988 of file `unordered_set.h`.

**5.1086.3.7 unordered\_multiset()** [7/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
    const allocator_type & __a ) [inline], [explicit]
```

Creates an `unordered_multiset` with no elements.

**Parameters**

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 1009 of file `unordered_set.h`.

**5.1086.4 Member Function Documentation****5.1086.4.1 begin()** [1/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_multiset`.

Definition at line 1116 of file `unordered_set.h`.

#### 5.1086.4.2 begin() [2/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 1120 of file unordered\_set.h.

#### 5.1086.4.3 begin() [3/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

$\_n$	The bucket index.
-------	-------------------

##### Returns

A read-only local iterator.

Definition at line 1502 of file unordered\_set.h.

#### 5.1086.4.4 begin() [4/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

$\_n$	The bucket index.
-------	-------------------

**Returns**

A read-only local iterator.

Definition at line 1506 of file unordered\_set.h.

**5.1086.4.5 bucket\_count()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_multiset.

Definition at line 1468 of file unordered\_set.h.

**5.1086.4.6 cbegin()** [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.


Definition at line 1143 of file unordered\_set.h.

**5.1086.4.7 cbegin()** [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

	The bucket index.
<code>__n</code>	

**Returns**

A read-only local iterator.

Definition at line 1510 of file unordered\_set.h.

**5.1086.4.8 cend()** [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

Definition at line 1151 of file unordered\_set.h.

**5.1086.4.9 cend()** [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

$\_n$	The bucket index.
-------	-------------------

**Returns**

A read-only local iterator.

Definition at line 1530 of file unordered\_set.h.

**5.1086.4.10 clear()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in an unordered\_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1352 of file `unordered_set.h`.

#### 5.1086.4.11 `count()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements.

##### Parameters

<code>__x</code>	Element to located.
------------------	---------------------

##### Returns

Number of elements with specified key.

Definition at line 1445 of file `unordered_set.h`.

#### 5.1086.4.12 `emplace()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Builds and insert an element into the `unordered_multiset`.

##### Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

##### Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 1165 of file unordered\_set.h.

#### 5.1086.4.13 `emplace_hint()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Inserts an element into the `unordered_multiset`.

##### Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

##### Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant time.

Definition at line 1187 of file unordered\_set.h.

#### 5.1086.4.14 `empty()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the `unordered_multiset` is empty.

Definition at line 1095 of file unordered\_set.h.

**5.1086.4.15** `end()` [1/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multiset`.

Definition at line 1130 of file `unordered_set.h`.

**5.1086.4.16** `end()` [2/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multiset`.

Definition at line 1134 of file `unordered_set.h`.

**5.1086.4.17** `end()` [3/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
    size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 1522 of file `unordered_set.h`.

**5.1086.4.18** end() [4/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

<code>_↵</code>	The bucket index.
<code>_n</code>	

**Returns**

A read-only local iterator.

Definition at line 1526 of file unordered\_set.h.

**5.1086.4.19** equal\_range() [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_↵
range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

**Parameters**

<code>_↵</code>	Key to be located.
<code>_x</code>	

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1456 of file unordered\_set.h.

**5.1086.4.20** equal\_range() [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```



```
std::pair<const_iterator, const_iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (
    const key_type & __x ) const    [inline]
```

Finds a subsequence matching given key.

#### Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1460 of file unordered\_set.h.

#### 5.1086.4.21 erase() [1/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position )    [inline]
```

Erases an element from an unordered\_multiset.

#### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

#### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1298 of file unordered\_set.h.

## 5.1086.4.22 erase() [2/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (  
    iterator __position ) [inline]
```

Erases an element from an unordered\_multiset.

**Parameters**

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1303 of file `unordered_set.h`.

**5.1086.4.23 erase()** [3/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (  
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

**Parameters**

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1321 of file `unordered_set.h`.

**5.1086.4.24** erase() [ 4/4 ]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (  
    const_iterator __first,  
    const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multiset`.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1341 of file `unordered_set.h`.

**5.1086.4.25 find() [1/2]**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_multiset`.

**Parameters**

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1431 of file `unordered_set.h`.

**5.1086.4.26 find() [2/2]**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_multiset`.

## Parameters

<code>_↵</code>	Element to be located.
<code>_X</code>	

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1435 of file `unordered_set.h`.

## 5.1086.4.27 get\_allocator()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator ( ) const
[inline], [noexcept]
```

Returns the allocator object used by the `unordered_multiset`.

Definition at line 1088 of file `unordered_set.h`.

## 5.1086.4.28 hash\_function()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]
```

Returns the hash functor object with which the `unordered_multiset` was constructed.


Definition at line 1407 of file `unordered_set.h`.

## 5.1086.4.29 insert() [1/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Inserts an element into the `unordered_multiset`.

**Parameters**

	Element to be inserted.
<code>_X</code>	

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.


Definition at line 1199 of file `unordered_set.h`.

**5.1086.4.30 insert()** [2/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (  
    value_type && __x ) [inline]
```

Inserts an element into the `unordered_multiset`.

**Parameters**

	Element to be inserted.
<code>_X</code>	

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 1203 of file `unordered_set.h`.

**5.1086.4.31 insert()** [3/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (  
    const_iterator __hint,  
    const value_type & __x ) [inline]
```

Inserts an element into the `unordered_multiset`.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

## Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 1225 of file unordered\_set.h.

## 5.1086.4.32 insert() [4/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x ) [inline]
```

Inserts an element into the unordered\_multiset.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

## Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 1229 of file unordered\_set.h.



**5.1086.4.33 insert()** [ 5/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that inserts a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 1243 of file unordered\_set.h.

**5.1086.4.34 insert()** [ 6/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Inserts a list of elements into the unordered\_multiset.

**Parameters**

<code>↩</code>	A std::initializer_list<value_type> of elements to be inserted.
<code>__</code> ↩	
<code>↩</code>	
<code>__</code> ↩	
<code>/</code>	

Complexity similar to that of the range constructor.

Definition at line 1254 of file unordered\_set.h.

**5.1086.4.35 key\_eq()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]
```

Returns the key comparison object with which the unordered\_multiset was constructed.

Definition at line 1413 of file unordered\_set.h.

#### 5.1086.4.36 load\_factor()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1538 of file unordered\_set.h.

#### 5.1086.4.37 max\_bucket\_count()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multiset.

Definition at line 1473 of file unordered\_set.h.

#### 5.1086.4.38 max\_load\_factor() [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multiset tries to keep the load factor less than or equal to.

Definition at line 1544 of file unordered\_set.h.

#### 5.1086.4.39 max\_load\_factor() [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z ) [inline]
```

Change the unordered\_multiset maximum load factor.

**Parameters**

<code>_Z</code>	The new maximum load factor.
-----------------	------------------------------

Definition at line 1552 of file `unordered_set.h`.

**5.1086.4.40 `max_size()`**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size ( ) const [inline],
[noexcept]
```

Returns the maximum size of the `unordered_multiset`.

Definition at line 1105 of file `unordered_set.h`.

**5.1086.4.41 `operator=()` [1/3]**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
    const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

**5.1086.4.42 `operator=()` [2/3]**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
    unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

**5.1086.4.43 `operator=()` [3/3]**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Unordered\_multiset list assignment operator.

## Parameters

$\leftrightarrow$	An initializer_list.
$\_ \leftrightarrow$	
$\leftrightarrow$	
$\_ \leftrightarrow$	
$\_ /$	

This function fills an unordered\_multiset with copies of the elements in the initializer list  $\_ /$ .

Note that the assignment completely changes the unordered\_multiset and that the resulting unordered\_multiset's size is the same as the number of elements assigned.

Definition at line 1080 of file unordered\_set.h.

## 5.1086.4.44 rehash()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (
    size_type __n ) [inline]
```

May rehash the unordered\_multiset.

## Parameters

$\_ \leftrightarrow$	The new number of buckets.
$\_ n$	

Rehash will occur only if the new number of buckets respect the unordered\_multiset maximum load factor.

Definition at line 1563 of file unordered\_set.h.

## 5.1086.4.45 reserve()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (
    size_type __n ) [inline]
```

Prepare the unordered\_multiset for a specified number of elements.

## Parameters

$\_ \leftrightarrow$	Number of elements required.
$\_ n$	

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1574 of file `unordered_set.h`.

#### 5.1086.4.46 `size()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size ( ) const [inline],
[noexcept]
```

Returns the size of the `unordered_multiset`.

Definition at line 1100 of file `unordered_set.h`.

#### 5.1086.4.47 `swap()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (
    unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another `unordered_multiset`.

##### Parameters

<code>__x</code>	An <code>unordered_multiset</code> of the same element and allocator types.
------------------	---

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 1365 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 5.1087 `std::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
  - typedef \_Hashtable::value\_type [value\\_type](#)
  - typedef \_Hashtable::hasher [hasher](#)
  - typedef \_Hashtable::key\_equal [key\\_equal](#)
  - typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- 
- typedef \_Hashtable::pointer [pointer](#)
  - typedef \_Hashtable::const\_pointer [const\\_pointer](#)
  - typedef \_Hashtable::reference [reference](#)
  - typedef \_Hashtable::const\_reference [const\\_reference](#)
  - typedef \_Hashtable::iterator [iterator](#)
  - typedef \_Hashtable::const\_iterator [const\\_iterator](#)
  - typedef \_Hashtable::local\_iterator [local\\_iterator](#)
  - typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
  - typedef \_Hashtable::size\_type [size\\_type](#)
  - typedef \_Hashtable::difference\_type [difference\\_type](#)

#### Public Member Functions

- [unordered\\_set](#) ()=default
- [unordered\\_set](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator >  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_set](#) (const [unordered\\_set](#) &)=default
- [unordered\\_set](#) ([unordered\\_set](#) &&)=default
- [unordered\\_set](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) (const [unordered\\_set](#) &\_\_uset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) ([unordered\\_set](#) &&\_\_uset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_set](#) ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [size\\_type](#) [bucket](#) (const [key\\_type](#) &\_\_key) const
- [size\\_type](#) [bucket\\_count](#) () const noexcept
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_n) const
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept

- void `clear` () noexcept
- `size_type count` (const `key_type` &\_\_x) const
- template<typename... \_Args>  
`std::pair`< `iterator`, bool > `emplace` (\_Args &&... \_\_args)
- template<typename... \_Args>  
`iterator emplace_hint` (const `iterator` \_\_pos, \_Args &&... \_\_args)
- bool `empty` () const noexcept
- `size_type erase` (const `key_type` &\_\_x)
- `iterator erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `allocator_type get_allocator` () const noexcept
- `hasher hash_function` () const
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `insert` (initializer\_list< `value_type` > \_\_l)
- `key_equal key_eq` () const
- float `load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_z)
- `size_type max_size` () const noexcept
- `unordered_set` & `operator=` (const `unordered_set` &)=default
- `unordered_set` & `operator=` (`unordered_set` &&)=default
- `unordered_set` & `operator=` (initializer\_list< `value_type` > \_\_l)
- void `rehash` (`size_type` \_\_n)
- void `reserve` (`size_type` \_\_n)
- `size_type size` () const noexcept
- void `swap` (`unordered_set` &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))

- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept

- `iterator end` () noexcept
- `const_iterator end` () const noexcept

- `std::pair`< `iterator`, bool > `insert` (const `value_type` &\_\_x)
- `std::pair`< `iterator`, bool > `insert` (`value_type` &&\_\_x)

- `iterator insert` (const `iterator` \_\_hint, const `value_type` &\_\_x)

- `iterator insert` (`const_iterator` \_\_hint, `value_type` &&\_\_x)
- `iterator erase` (`const_iterator` \_\_position)
- `iterator erase` (`iterator` \_\_position)
- `iterator find` (`const key_type` &\_\_x)
- `const_iterator find` (`const key_type` &\_\_x) const
- `std::pair< iterator, iterator > equal_range` (`const key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type` &\_\_x) const
- `local_iterator begin` (`size_type` \_\_n)
- `const_local_iterator begin` (`size_type` \_\_n) const
- `const_local_iterator cbegin` (`size_type` \_\_n) const
- `local_iterator end` (`size_type` \_\_n)
- `const_local_iterator end` (`size_type` \_\_n) const
- `const_local_iterator cend` (`size_type` \_\_n) const

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (`const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &`, `const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &`)

#### 5.1087.1 Detailed Description

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc =
allocator<_Value>>
class std::unordered_set< _Value, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.



### Template Parameters

<code>_Value</code>	Type of key objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__uset_hashtable`.

Definition at line 97 of file `unordered_set.h`.

## 5.1087.2 Member Typedef Documentation

### 5.1087.2.1 `allocator_type`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::allocator_type std::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc>::allocator\_type
```

Public typedefs.

Definition at line 110 of file `unordered_set.h`.

### 5.1087.2.2 `const_iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::const_iterator std::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc>::const\_iterator
```

Iterator-related typedefs.

Definition at line 120 of file `unordered_set.h`.

### 5.1087.2.3 `const_local_iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::const_local_iterator std::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc>↵  
::const\_local\_iterator
```

Iterator-related typedefs.

Definition at line 122 of file `unordered_set.h`.

#### 5.1087.2.4 const\_pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::const\_pointer
```

Iterator-related typedefs.

Definition at line 116 of file unordered\_set.h.

#### 5.1087.2.5 const\_reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::const\_reference
```

Iterator-related typedefs.

Definition at line 118 of file unordered\_set.h.

#### 5.1087.2.6 difference\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::difference\_type
```

Iterator-related typedefs.

Definition at line 124 of file unordered\_set.h.

#### 5.1087.2.7 hasher

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 108 of file unordered\_set.h.

#### 5.1087.2.8 iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 119 of file unordered\_set.h.

#### 5.1087.2.9 key\_equal

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 109 of file unordered\_set.h.

#### 5.1087.2.10 key\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 106 of file unordered\_set.h.

#### 5.1087.2.11 local\_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered\_set< _Value, _Hash, _Pred, _Alloc >::local_iterator
```

Iterator-related typedefs.

Definition at line 121 of file unordered\_set.h.

#### 5.1087.2.12 pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::pointer std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 115 of file unordered\_set.h.

#### 5.1087.2.13 reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::reference std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >::reference
```

Iterator-related typedefs.

Definition at line 117 of file unordered\_set.h.

#### 5.1087.2.14 size\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::size_type std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >::size\_type
```

Iterator-related typedefs.

Definition at line 123 of file unordered\_set.h.

#### 5.1087.2.15 value\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,  
typename _Alloc = allocator<_Value>>  
typedef _Hashtable::value_type std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >::value\_type
```

Public typedefs.

Definition at line 107 of file unordered\_set.h.

### 5.1087.3 Constructor & Destructor Documentation

**5.1087.3.1 unordered\_set()** [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set ( ) [default]
```

Default constructor.

**5.1087.3.2 unordered\_set()** [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    size_type __n,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Default constructor creates no elements.

**Parameters**

<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq1</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 145 of file unordered\_set.h.

**5.1087.3.3 unordered\_set()** [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    _InputIterator __first,
    _InputIterator __last,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_set from a range.

**Parameters**

<code>__first</code>	An input iterator.
----------------------	--------------------

## Parameters

<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_set` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 166 of file `unordered_set.h`.

## 5.1087.3.4 unordered\_set() [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    const unordered_set< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

## 5.1087.3.5 unordered\_set() [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    unordered_set< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

## 5.1087.3.6 unordered\_set() [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    const allocator_type & __a ) [inline], [explicit]
```

Creates an `unordered_set` with no elements.

## Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 185 of file unordered\_set.h.

### 5.1087.3.7 unordered\_set() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
    initializer_list< value_type > __l,
    size_type __n = 0,
    const hasher & __hf = hasher(),
    const key_equal & __eq1 = key_equal(),
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered\_set from an initializer\_list.

#### Parameters

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eq1</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered\_set consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 220 of file unordered\_set.h.

## 5.1087.4 Member Function Documentation

### 5.1087.4.1 begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 319 of file unordered\_set.h.

#### 5.1087.4.2 begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 323 of file unordered\_set.h.

#### 5.1087.4.3 begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

$\_n$	The bucket index.
-------	-------------------

##### Returns

A read-only local iterator.

Definition at line 726 of file unordered\_set.h.

#### 5.1087.4.4 begin() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

##### Parameters

$\_n$	The bucket index.
-------	-------------------



**Returns**

A read-only local iterator.

Definition at line 730 of file unordered\_set.h.

**5.1087.4.5 bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_set.

Definition at line 692 of file unordered\_set.h.

**5.1087.4.6 cbegin()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 346 of file unordered\_set.h.

**5.1087.4.7 cbegin()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 734 of file unordered\_set.h.

**5.1087.4.8 cend()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend ( ) const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

Definition at line 354 of file unordered\_set.h.

**5.1087.4.9 cend()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 754 of file unordered\_set.h.

**5.1087.4.10 clear()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all elements in an `unordered_set`. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 572 of file `unordered_set.h`.

#### 5.1087.4.11 `count()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (
    const key_type & __x ) const [inline]
```

Finds the number of elements.

##### Parameters

<code>__x</code>	Element to located.
------------------	---------------------

##### Returns

Number of elements with specified key.

This function only makes sense for `unordered_multisets`; for `unordered_set` the result will either be 0 (not present) or 1 (present).

Definition at line 667 of file `unordered_set.h`.

#### 5.1087.4.12 `emplace()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace (
    _Args &&... __args ) [inline]
```

Attempts to build and insert an element into the `unordered_set`.

##### Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 376 of file unordered\_set.h.

**5.1087.4.13   emplace\_hint()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
    const_iterator __pos,
    _Args &&... __args ) [inline]
```

Attempts to insert an element into the unordered\_set.

**Parameters**

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant time.

Definition at line 402 of file unordered\_set.h.

**5.1087.4.14** `empty()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]
```

Returns true if the `unordered_set` is empty.

Definition at line 298 of file `unordered_set.h`.

**5.1087.4.15** `end()` [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 333 of file `unordered_set.h`.

**5.1087.4.16** `end()` [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 337 of file `unordered_set.h`.

**5.1087.4.17** `end()` [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
    size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

<code>__n</code>	The bucket index.
------------------	-------------------

**Returns**

A read-only local iterator.

Definition at line 746 of file unordered\_set.h.

**5.1087.4.18 end()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
    size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

$\_n$	The bucket index.
-------	-------------------

**Returns**

A read-only local iterator.

Definition at line 750 of file unordered\_set.h.

**5.1087.4.19 equal\_range()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, iterator> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (
    const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

**Parameters**

$\_x$	Key to be located.
-------	--------------------

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 680 of file unordered\_set.h.

#### 5.1087.4.20 equal\_range() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<const_iterator, const_iterator> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::
::equal_range (
    const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

##### Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

##### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 684 of file unordered\_set.h.

#### 5.1087.4.21 erase() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __position ) [inline]
```

Erases an element from an unordered\_set.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

##### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 522 of file unordered\_set.h.

#### 5.1087.4.22 erase() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    iterator __position ) [inline]
```

Erases an element from an unordered\_set.

##### Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

##### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 527 of file unordered\_set.h.

#### 5.1087.4.23 erase() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    const key_type & __x ) [inline]
```

Erases elements according to the provided key.

##### Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

##### Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_set. For an unordered\_set the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that



if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 544 of file `unordered_set.h`.

#### 5.1087.4.24 `erase()` [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_set`.

##### Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

##### Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 562 of file `unordered_set.h`.

#### 5.1087.4.25 `find()` [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_set`.

##### Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 649 of file unordered\_set.h.

**5.1087.4.26 find()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
    const key_type & __x ) const [inline]
```

Tries to locate an element in an unordered\_set.

**Parameters**

$\_x$	Element to be located.
-------	------------------------

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

Definition at line 653 of file unordered\_set.h.

**5.1087.4.27 get\_allocator()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator ( ) const [inline],
[noexcept]
```

Returns the allocator object used by the unordered\_set.

Definition at line 291 of file unordered\_set.h.

**5.1087.4.28** `hash_function()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]
```

Returns the hash functor object with which the `unordered_set` was constructed.

Definition at line 625 of file `unordered_set.h`.

**5.1087.4.29** `insert()` [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    const value_type & __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

<code>__x</code>	Element to be inserted.
------------------	-------------------------

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

Definition at line 420 of file `unordered_set.h`.

**5.1087.4.30** `insert()` [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    value_type && __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

## Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 424 of file unordered\_set.h.

## 5.1087.4.31 insert() [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    const value_type & __x ) [inline]
```

Attempts to insert an element into the unordered\_set.

## Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

## Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 449 of file unordered\_set.h.

**5.1087.4.32 insert()** [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    const_iterator __hint,
    value_type && __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 453 of file `unordered_set.h`.

**5.1087.4.33 insert()** [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 468 of file unordered\_set.h.

#### 5.1087.4.34 insert() [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
    initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the unordered\_set.

##### Parameters

↩	A std::initializer_list<value_type> of elements to be inserted.
↩	
↩	
↩	
↩	
↩	

Complexity similar to that of the range constructor.

Definition at line 479 of file unordered\_set.h.

#### 5.1087.4.35 key\_eq()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]
```

Returns the key comparison object with which the unordered\_set was constructed.

Definition at line 631 of file unordered\_set.h.

#### 5.1087.4.36 load\_factor()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 762 of file unordered\_set.h.

**5.1087.4.37** `max_bucket_count()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline],
[noexcept]
```

Returns the maximum number of buckets of the `unordered_set`.

Definition at line 697 of file `unordered_set.h`.

**5.1087.4.38** `max_load_factor()` [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline],
[noexcept]
```

Returns a positive number that the `unordered_set` tries to keep the load factor less than or equal to.

Definition at line 768 of file `unordered_set.h`.

**5.1087.4.39** `max_load_factor()` [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
    float __z ) [inline]
```

Change the `unordered_set` maximum load factor.

**Parameters**

<code>__z</code>	The new maximum load factor.
------------------	------------------------------

Definition at line 776 of file `unordered_set.h`.

**5.1087.4.40** `max_size()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the maximum size of the unordered\_set.

Definition at line 308 of file unordered\_set.h.

#### 5.1087.4.41 operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
    const unordered_set< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

#### 5.1087.4.42 operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
    unordered_set< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

#### 5.1087.4.43 operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Unordered\_set list assignment operator.

##### Parameters

↩	An initializer_list.
↩	
↩	
↩	
/	

This function fills an unordered\_set with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_set and that the resulting unordered\_set's size is the same as the number of elements assigned.



Definition at line 283 of file unordered\_set.h.

#### 5.1087.4.44 rehash()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (
    size_type __n ) [inline]
```

May rehash the unordered\_set.

##### Parameters

<code><math>\leftarrow</math></code> <code>__n</code>	The new number of buckets.
--	----------------------------

Rehash will occur only if the new number of buckets respect the unordered\_set maximum load factor.

Definition at line 787 of file unordered\_set.h.

#### 5.1087.4.45 reserve()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (
    size_type __n ) [inline]
```

Prepare the unordered\_set for a specified number of elements.

##### Parameters

<code><math>\leftarrow</math></code> <code>__n</code>	Number of elements required.
--	------------------------------

Same as rehash(ceil(n / max\_load\_factor())).

Definition at line 798 of file unordered\_set.h.

#### 5.1087.4.46 size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the size of the unordered\_set.

Definition at line 303 of file unordered\_set.h.

#### 5.1087.4.47 swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (
    unordered_set< _Value, _Hash, _Pred, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another unordered\_set.

#### Parameters

<code>__x</code>	An unordered_set of the same element and allocator types.
------------------	---

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

Definition at line 585 of file unordered\_set.h.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 5.1088 std::uses\_allocator<\_Tp, \_Alloc > Struct Template Reference

Inherits type<\_Tp, \_Alloc >.

### 5.1088.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::uses_allocator<_Tp, _Alloc >
```

Declare uses\_allocator so it can be specialized in <queue> etc.

[allocator.uses.trait]

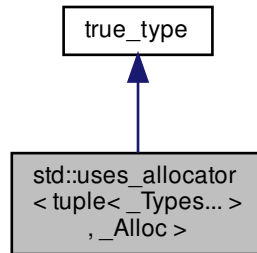
Definition at line 72 of file memoryfwd.h.

The documentation for this struct was generated from the following file:

- [memoryfwd.h](#)

## 5.1089 std::uses\_allocator< tuple< \_Types... >, \_Alloc > Struct Template Reference

Inheritance diagram for std::uses\_allocator< tuple< \_Types... >, \_Alloc >:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

### 5.1089.1 Detailed Description

```
template<typename... _Types, typename _Alloc>
struct std::uses_allocator< tuple< _Types... >, _Alloc >
```

Partial specialization for tuples.

Definition at line 1646 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.1090 std::valarray&lt; \_Tp &gt; Class Template Reference

## Public Types

- typedef \_Tp **value\_type**

## Public Member Functions

- [valarray](#) ()
- [valarray](#) (size\_t)
- [valarray](#) (const \_Tp &, size\_t)
- [valarray](#) (const \_Tp \* \_\_restrict\_\_, size\_t)
- [valarray](#) (const [valarray](#) &)
- [valarray](#) ([valarray](#) &&) noexcept
- [valarray](#) (const [slice\\_array](#)< \_Tp > &)
- [valarray](#) (const [gslice\\_array](#)< \_Tp > &)
- [valarray](#) (const [mask\\_array](#)< \_Tp > &)
- [valarray](#) (const [indirect\\_array](#)< \_Tp > &)
- [valarray](#) ([initializer\\_list](#)< \_Tp >)
- template<class \_Dom >  
**valarray** (const \_Expr< \_Dom, \_Tp > &\_\_e)
- template<typename \_Tp>  
**valarray** (const \_Tp \* \_\_restrict\_\_ \_\_p, size\_t \_\_n)
- \_Expr< \_ValFunClos< \_ValArray, \_Tp >, \_Tp > [apply](#) (\_Tp func(\_Tp)) const
- \_Expr< \_RefFunClos< \_ValArray, \_Tp >, \_Tp > [apply](#) (\_Tp func(const \_Tp &)) const
- [valarray](#)< \_Tp > [cshift](#) (int \_\_n) const
- \_Tp [max](#) () const
- \_Tp [min](#) () const
- \_UnaryOp< \_\_logical\_not >::\_Rt [operator!](#) () const
- [valarray](#)< \_Tp > & [operator%=>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator%=>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator%=>](#) (const \_Expr< \_Dom, \_Tp > &)
- [valarray](#)< \_Tp > & [operator&=>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator&=>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator&=>](#) (const \_Expr< \_Dom, \_Tp > &)
- [valarray](#)< \_Tp > & [operator\\*=>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator\\*=>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator\\*=>](#) (const \_Expr< \_Dom, \_Tp > &)
- \_UnaryOp< \_\_unary\_plus >::\_Rt [operator+>](#) () const
- [valarray](#)< \_Tp > & [operator+>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator+>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator+>](#) (const \_Expr< \_Dom, \_Tp > &)
- \_UnaryOp< \_\_negate >::\_Rt [operator->](#) () const
- [valarray](#)< \_Tp > & [operator->](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator->](#) (const [valarray](#)< \_Tp > &)

- `template<class _Dom >`  
`valarray<_Tp> & operator-= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator/= (const _Tp &)`
- `valarray<_Tp> & operator/= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator/= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator<=<= (const _Tp &)`
- `valarray<_Tp> & operator<=<= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator<<= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator= (const valarray<_Tp> &__v)`
- `valarray<_Tp> & operator= (valarray<_Tp> &&__v) noexcept`
- `valarray<_Tp> & operator= (const _Tp &__t)`
- `valarray<_Tp> & operator= (const slice_array<_Tp> &__sa)`
- `valarray<_Tp> & operator= (const gslice_array<_Tp> &__ga)`
- `valarray<_Tp> & operator= (const mask_array<_Tp> &__ma)`
- `valarray<_Tp> & operator= (const indirect_array<_Tp> &__ia)`
- `valarray & operator= (initializer_list<_Tp> __l)`
- `template<class _Dom >`  
`valarray<_Tp> & operator= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator>>= (const _Tp &)`
- `valarray<_Tp> & operator>>= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator>>= (const _Expr<_Dom, _Tp> &)`
- `_Tp & operator[] (size_t __i)`
- `const _Tp & operator[] (size_t) const`
- `_Expr<_SClos<_ValArray, _Tp>, _Tp> operator[] (slice __s) const`
- `slice_array<_Tp> operator[] (slice __s)`
- `_Expr<_GClos<_ValArray, _Tp>, _Tp> operator[] (const gslice &__s) const`
- `gslice_array<_Tp> operator[] (const gslice &__s)`
- `valarray<_Tp> operator[] (const valarray<bool> &__m) const`
- `mask_array<_Tp> operator[] (const valarray<bool> &__m)`
- `_Expr<_IClos<_ValArray, _Tp>, _Tp> operator[] (const valarray<size_t> &__i) const`
- `indirect_array<_Tp> operator[] (const valarray<size_t> &__i)`
- `valarray<_Tp> & operator^= (const _Tp &)`
- `valarray<_Tp> & operator^= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator^= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator|= (const _Tp &)`
- `valarray<_Tp> & operator|= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator|= (const _Expr<_Dom, _Tp> &)`
- `_UnaryOp<__bitwise_not>::Rt operator~ () const`
- `void resize (size_t __size, _Tp __c=_Tp())`
- `valarray<_Tp> shift (int __n) const`
- `size_t size () const`
- `_Tp sum () const`
- `void swap (valarray<_Tp> &__v) noexcept`

## Friends

- `class _Array<_Tp>`

## 5.1090.1 Detailed Description

```
template<class _Tp>
class std::valarray<_Tp>
```

Smart array designed to support numeric processing.

A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

## Template Parameters

<code>_Tp</code>	Type of object in the array.
------------------	------------------------------

Definition at line 78 of file valarray.

## 5.1090.2 Constructor &amp; Destructor Documentation

5.1090.2.1 `valarray()`

```
template<class _Tp>
std::valarray<_Tp>::valarray (
    const _Tp * __restrict__,
    size_t )
```

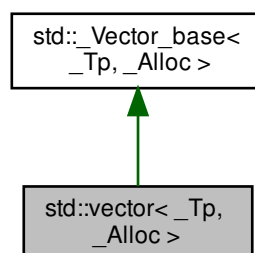
Construct an array initialized to the first  $n$  elements of  $t$ .

The documentation for this class was generated from the following file:

- [valarray](#)

5.1091 `std::vector<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::vector<_Tp, _Alloc>`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [vector](#) () noexcept([is\\_nothrow\\_default\\_constructible](#)< `_Alloc` >::value)
- [vector](#) (const allocator\_type &\_\_a) noexcept
- [vector](#) (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- [vector](#) (size\_type \_\_n, const value\_type &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- [vector](#) (const [vector](#) &\_\_x)
- [vector](#) ([vector](#) &&\_\_x) noexcept
- [vector](#) (const [vector](#) &\_\_x, const allocator\_type &\_\_a)
- [vector](#) ([vector](#) &&\_\_rv, const allocator\_type &\_\_m) noexcept(`_Alloc_traits::S_always_equal()`)
- [vector](#) ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename `_InputIterator` , typename = `std::_RequireInputIter<_InputIterator>>`  
  [vector](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const allocator\_type &\_\_a=allocator\_type())
- [~vector](#) () noexcept
- template<typename... `_Args`>  
  auto **`_M_emplace_aux`** (const\_iterator \_\_position, `_Args` &&... \_\_args) -> iterator
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename `_InputIterator` , typename = `std::_RequireInputIter<_InputIterator>>`  
  void [assign](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void [assign](#) ([initializer\\_list](#)< value\_type > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) () noexcept
- const\_reference [back](#) () const noexcept
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- [const\\_reverse\\_iterator](#) [crbegin](#) () const noexcept
- [const\\_reverse\\_iterator](#) [crend](#) () const noexcept
- `_Tp` \* [data](#) () noexcept
- const `_Tp` \* **`data`** () const noexcept

- `template<typename... _Args>`  
`iterator` **emplace** (`const_iterator` \_\_position, `_Args` &&... \_\_args)
- `template<typename... _Args>`  
`void` **emplace\_back** (`_Args` &&... \_\_args)
- `bool` **empty** () `const` `noexcept`
- `iterator` **end** () `noexcept`
- `const_iterator` **end** () `const` `noexcept`
- `iterator` **erase** (`const_iterator` \_\_position)
- `iterator` **erase** (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `reference` **front** () `noexcept`
- `const_reference` **front** () `const` `noexcept`
- `allocator_type` **get\_allocator** () `const` `noexcept`
- `iterator` **insert** (`const_iterator` \_\_position, `const value_type` &\_\_x)
- `iterator` **insert** (`const_iterator` \_\_position, `value_type` &&\_\_x)
- `iterator` **insert** (`const_iterator` \_\_position, `initializer_list`< `value_type` > \_\_l)
- `iterator` **insert** (`const_iterator` \_\_position, `size_type` \_\_n, `const value_type` &\_\_x)
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`iterator` **insert** (`const_iterator` \_\_position, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `size_type` **max\_size** () `const` `noexcept`
- `vector` & **operator=** (`const vector` &\_\_x)
- `vector` & **operator=** (`vector` &&\_\_x) `noexcept` (`_Alloc_traits::S_nothrow_move`())
- `vector` & **operator=** (`initializer_list`< `value_type` > \_\_l)
- `reference` **operator[]** (`size_type` \_\_n) `noexcept`
- `const_reference` **operator[]** (`size_type` \_\_n) `const` `noexcept`
- `void` **pop\_back** () `noexcept`
- `void` **push\_back** (`const value_type` &\_\_x)
- `void` **push\_back** (`value_type` &&\_\_x)
- `reverse_iterator` **rbegin** () `noexcept`
- `const_reverse_iterator` **rbegin** () `const` `noexcept`
- `reverse_iterator` **rend** () `noexcept`
- `const_reverse_iterator` **rend** () `const` `noexcept`
- `void` **reserve** (`size_type` \_\_n)
- `void` **resize** (`size_type` \_\_new\_size)
- `void` **resize** (`size_type` \_\_new\_size, `const value_type` &\_\_x)
- `void` **shrink\_to\_fit** ()
- `size_type` **size** () `const` `noexcept`
- `void` **swap** (`vector` &\_\_x) `noexcept`

#### Protected Member Functions

- `pointer` **\_M\_allocate** (`size_t` \_\_n)
- `pointer` **\_M\_allocate** (`size_t` \_\_n)
- `template<typename _ForwardIterator>`  
`pointer` **\_M\_allocate\_and\_copy** (`size_type` \_\_n, `_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last)
- `template<typename _InputIterator>`  
`void` **\_M\_assign\_aux** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `std::input_iterator_tag`)
- `template<typename _ForwardIterator>`  
`void` **\_M\_assign\_aux** (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `std::forward_iterator_tag`)
- `template<typename _Integer>`  
`void` **\_M\_assign\_dispatch** (`_Integer` \_\_n, `_Integer` \_\_val, `_true_type`)



- `template<typename _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`
- `void _M_deallocate (pointer __p, size_t __n)`
- `void _M_deallocate (pointer __p, size_t __n)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize (size_type __n)`
- `template<typename... _Args>`  
`iterator _M_emplace_aux (const_iterator __position, _Args &&... __args)`
- `iterator _M_emplace_aux (const_iterator __position, value_type &&__v)`
- `iterator _M_erase (iterator __position)`
- `iterator _M_erase (iterator __first, iterator __last)`
- `void _M_erase_at_end (pointer __pos) noexcept`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __value, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _Arg >`  
`void _M_insert_aux (iterator __position, _Arg &&__arg)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `iterator _M_insert_rval (const_iterator __position, value_type &&__v)`
- `void _M_range_check (size_type __n) const`
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator >`  
`void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_insert (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename... _Args>`  
`void _M_realloc_insert (iterator __position, _Args &&... __args)`
- `bool _M_shrink_to_fit ()`
- `allocator_type get_allocator () const noexcept`

## Protected Attributes

- `_Vector_impl _M_impl`

## 5.1091.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::vector<_Tp, _Alloc>
```

A standard container which offers fixed time access to individual elements in any order.

## Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 339 of file `stl_vector.h`.

## 5.1091.2 Constructor &amp; Destructor Documentation

5.1091.2.1 `vector()` [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector ( ) [inline], [noexcept]
```

Creates a vector with no elements.

Definition at line 391 of file `stl_vector.h`.

5.1091.2.2 `vector()` [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
    const allocator_type & __a ) [inline], [explicit], [noexcept]
```

Creates a vector with no elements.

## Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 402 of file `stl_vector.h`.

#### 5.1091.2.3 `vector()` [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    size_type __n,
    const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a vector with default constructed elements.

##### Parameters

<code>__n</code>	The number of elements to initially create.
<code>__a</code>	An allocator.

This constructor fills the vector with `__n` default constructed elements.

Definition at line 415 of file `stl_vector.h`.

#### 5.1091.2.4 `vector()` [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    size_type __n,
    const value_type & __value,
    const allocator_type & __a = allocator_type() ) [inline]
```

Creates a vector with copies of an exemplar element.

##### Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator.

This constructor fills the vector with `__n` copies of `__value`.


Definition at line 427 of file `stl_vector.h`.

## 5.1091.2.5 vector() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    const vector< _Tp, _Alloc > & __x ) [inline]
```

Vector copy constructor.

## Parameters

	A vector of identical element and allocator types.
<code>__x</code>	

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied (i.e. `capacity() == size()` in the new vector).

The newly-created vector uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).


Definition at line 458 of file `stl_vector.h`.

## 5.1091.2.6 vector() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    vector< _Tp, _Alloc > && __x ) [inline], [noexcept]
```

Vector move constructor.

## Parameters

	A vector of identical element and allocator types.
<code>__x</code>	

The newly-created vector contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified vector.

Definition at line 476 of file `stl_vector.h`.

## 5.1091.2.7 vector() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    const vector< _Tp, _Alloc > & __x,
    const allocator_type & __a ) [inline]
```

Copy constructor with alternative allocator.

Definition at line 480 of file `stl_vector.h`.

**5.1091.2.8** `vector()` [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    vector< _Tp, _Alloc > && __rv,
    const allocator_type & __m ) [inline], [noexcept]
```

Move constructor with alternative allocator.

Definition at line 490 of file `stl_vector.h`.

**5.1091.2.9** `vector()` [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
    initializer_list< value_type > __l,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a vector from an initializer list.

**Parameters**

<code>__l</code>	An initializer_list.
<code>__a</code>	An allocator.

Create a vector consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 515 of file `stl_vector.h`.

**5.1091.2.10** `vector()` [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::RequireInputIter<_InputIterator>>
std::vector< _Tp, _Alloc >::vector (
    _InputIterator __first,
    _InputIterator __last,
    const allocator_type & __a = allocator_type() ) [inline]
```

Builds a vector from a range.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator.

Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

Definition at line 543 of file `std_vector.h`.

#### 5.1091.2.11 `~vector()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::~~vector ( ) [inline], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 565 of file `std_vector.h`.

### 5.1091.3 Member Function Documentation

#### 5.1091.3.1 `_M_allocate_and_copy()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _ForwardIterator >
pointer std::vector<_Tp, _Alloc>::_M_allocate_and_copy (
    size_type __n,
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

Definition at line 1395 of file `std_vector.h`.

#### 5.1091.3.2 `_M_range_check()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::_M_range_check (
    size_type __n ) const [inline], [protected]
```

Safety check used only from `at()`.

Definition at line 957 of file `std_vector.h`.

Referenced by `std::vector< sub_match<_Bi_iter>, allocator< sub_match<_Bi_iter>>>::at()`.

#### 5.1091.3.3 `assign()` [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::assign (
    size_type __n,
    const value_type & __val ) [inline]
```

Assigns a given value to a vector.

**Parameters**

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

Definition at line 636 of file `stl_vector.h`.

**5.1091.3.4 `assign()`** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::vector<_Tp, _Alloc>::assign (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Assigns a range to a vector.

**Parameters**

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a vector with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

Definition at line 655 of file `stl_vector.h`.

**5.1091.3.5 `assign()`** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::assign (
    initializer_list<value_type> __l ) [inline]
```

Assigns an initializer list to a vector.

## Parameters

<code>↩</code>	An initializer_list.
<code>↩</code>	
<code>↩</code>	
<code>↩</code>	
<code>/</code>	

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

Definition at line 681 of file `stl_vector.h`.

5.1091.3.6 `at()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::at (
    size_type __n ) [inline]
```

Provides access to the data contained in the vector.

## Parameters

<code>↩</code>	The index of the element for which data should be accessed.
<code>__n</code>	

## Returns

Read/write reference to data.

## Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 979 of file `stl_vector.h`.

5.1091.3.7 `at()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::at (
    size_type __n ) const [inline]
```



Provides access to the data contained in the vector.

## Parameters

<code>↵</code> <code>_n</code>	The index of the element for which data should be accessed.
-----------------------------------	---

## Returns

Read-only (constant) reference to data.

## Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 997 of file `stl_vector.h`.

5.1091.3.8 `back()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::back ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

Definition at line 1030 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution<_RealType>::max()`, and `std::piecewise_linear_distribution<↵_RealType>::max()`.

5.1091.3.9 `back()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::back ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 1041 of file `stl_vector.h`.

**5.1091.3.10 begin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
iterator std::vector< _Tp, _Alloc >::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 698 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::crend()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::erase()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`, `std::operator<()`, `std::operator==()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rend()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector()`.

**5.1091.3.11 begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
const_iterator std::vector< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 707 of file `stl_vector.h`.

**5.1091.3.12 capacity()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
size_type std::vector< _Tp, _Alloc >::capacity ( ) const [inline], [noexcept]
```

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 885 of file `stl_vector.h`.

**5.1091.3.13 cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
const_iterator std::vector< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 771 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::erase()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`.

#### 5.1091.3.14 `cend()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 780 of file `stl_vector.h`.

#### 5.1091.3.15 `clear()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::clear ( ) [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1385 of file `stl_vector.h`.

#### 5.1091.3.16 `crbegin()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 789 of file `stl_vector.h`.

#### 5.1091.3.17 `crend()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 798 of file `stl_vector.h`.

**5.1091.3.18 data()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp* std::vector<_Tp, _Alloc>::data ( ) [inline], [noexcept]
```

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

Definition at line 1055 of file stl\_vector.h.

Referenced by std::regex\_traits<\_CharType>::transform\_primary().

**5.1091.3.19 emplace()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
iterator std::vector<_Tp, _Alloc>::emplace (
    const_iterator __position,
    _Args &&... __args ) [inline]
```

Inserts an object in vector before specified iterator.

**Parameters**

<code>__position</code>	A const_iterator into the vector.
<code>__args</code>	Arguments.

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1135 of file stl\_vector.h.

**5.1091.3.20 empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector<_Tp, _Alloc>::empty ( ) const [inline], [noexcept]
```

Returns true if the vector is empty. (Thus begin() would equal end().)

Definition at line 894 of file stl\_vector.h.

Referenced by std::piecewise\_constant\_distribution<\_RealType>::densities(), std::piecewise\_linear\_distribution<\_RealType>::densities(), std::piecewise\_constant\_distribution<\_RealType>::intervals(), std::piecewise\_linear\_distribution<\_RealType>::intervals(), std::discrete\_distribution<\_IntType>::max(), std::piecewise\_constant\_distribution<\_RealType>::max(), std::piecewise\_linear\_distribution<\_RealType>::max(), std::piecewise\_constant\_distribution<\_RealType>::min(), std::piecewise\_linear\_distribution<\_RealType>::min(), and std::discrete\_distribution<\_IntType>::probabilities().

**5.1091.3.21 end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::end ( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 716 of file `std_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::cbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty()`, `std::operator<()`, `std::operator==()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector()`.

**5.1091.3.22 end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::end ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 725 of file `std_vector.h`.

**5.1091.3.23 erase()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
    const_iterator __position ) [inline]
```

Remove element at given position.

**Parameters**

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

**Returns**

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1317 of file `stl_vector.h`.

#### 5.1091.3.24 `erase()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
    const_iterator __first,
    const_iterator __last ) [inline]
```

Remove a range of elements.

##### Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

##### Returns

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1344 of file `stl_vector.h`.

#### 5.1091.3.25 `front()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front ( ) [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

Definition at line 1008 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution< _RealType >::min()`, and `std::piecewise_linear_distribution< _RealType >::min()`.

**5.1091.3.26** `front()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::front ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 1019 of file `std_vector.h`.

**5.1091.3.27** `get_allocator()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::_Vector_base<_Tp, _Alloc>::get_allocator [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 245 of file `std_vector.h`.

**5.1091.3.28** `insert()` [1/5]

```
template<typename _Tp , typename _Alloc >
vector<_Tp, _Alloc>::iterator vector::insert (
    const_iterator __position,
    const value_type & __x )
```

Inserts given value into vector before specified iterator.

**Parameters**

<code>__position</code>	A <code>const_iterator</code> into the vector.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 120 of file `vector.tcc`.



**5.1091.3.29** `insert()` [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
    const_iterator __position,
    value_type && __x ) [inline]
```

Inserts given rvalue into vector before specified iterator.

**Parameters**

<code>__position</code>	A const_iterator into the vector.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1180 of file `stl_vector.h`.

**5.1091.3.30** `insert()` [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
    const_iterator __position,
    initializer_list< value_type > __l ) [inline]
```

Inserts an `initializer_list` into the vector.

**Parameters**

<code>__position</code>	An iterator into the vector.
<code>__l</code>	An <code>initializer_list</code> .

This function will insert copies of the data in the `initializer_list l` into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1197 of file `stl_vector.h`.

**5.1091.3.31 insert()** [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
    const_iterator __position,
    size_type __n,
    const value_type & __x ) [inline]
```

Inserts a number of copies of given data into the vector.

**Parameters**

<code>__position</code>	A const_iterator into the vector.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1222 of file `stl_vector.h`.

**5.1091.3.32 insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>>
iterator std::vector< _Tp, _Alloc >::insert (
    const_iterator __position,
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Inserts a range into the vector.

**Parameters**

<code>__position</code>	A const_iterator into the vector.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1266 of file `stl_vector.h`.

**5.1091.3.33 `max_size()`**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]
```

Returns the `size()` of the largest possible vector.

Definition at line 810 of file `stl_vector.h`.

**5.1091.3.34 `operator=()` [1/3]**

```
template<typename _Tp , typename _Alloc >
vector< _Tp, _Alloc > & vector::operator= (
    const vector< _Tp, _Alloc > & __x )
```

Vector assignment operator.

**Parameters**

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied.

Whether the allocator is copied depends on the allocator traits.

Definition at line 187 of file `vector.tcc`.

**5.1091.3.35 `operator=()` [2/3]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector& std::vector< _Tp, _Alloc >::operator= (
    vector< _Tp, _Alloc > && __x ) [inline], [noexcept]
```

Vector move assignment operator.

## Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this vector (without copying, if the allocators permit it). Afterwards `__x` is a valid, but unspecified vector.

Whether the allocator is moved depends on the allocator traits.

Definition at line 596 of file `std_vector.h`.

## 5.1091.3.36 operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector& std::vector< _Tp, _Alloc >::operator= (
    initializer_list< value_type > __l ) [inline]
```

Vector list assignment operator.

## Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.


Definition at line 617 of file `std_vector.h`.

## 5.1091.3.37 operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (
    size_type __n ) [inline], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

 <code>_n</code>	The index of the element for which data should be accessed.
--	---

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)


Definition at line 930 of file `stl_vector.h`.

**5.1091.3.38 operator[]()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::operator[] (
    size_type __n ) const [inline], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

 <code>_n</code>	The index of the element for which data should be accessed.
--	---

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 948 of file `stl_vector.h`.

**5.1091.3.39 pop\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back ( ) [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1112 of file `stl_vector.h`.

5.1091.3.40 `push_back()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::push_back (
    const value_type & __x ) [inline]
```

Add data to the end of the vector.

## Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 1074 of file `std_vector.h`.

5.1091.3.41 `rbegin()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 734 of file `std_vector.h`.

5.1091.3.42 `rbegin()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 743 of file `std_vector.h`.

5.1091.3.43 `rend()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 752 of file `std_vector.h`.

**5.1091.3.44** `rend()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 761 of file `stl_vector.h`.

**5.1091.3.45** `reserve()`

```
template<typename _Tp , typename _Alloc >
void vector::reserve (
    size_type __n )
```

Attempt to preallocate enough memory for specified number of elements.

**Parameters**

<code>__n</code>	Number of elements required.
------------------	------------------------------

**Exceptions**

<code>std::length_error</code>	If <code>n</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 67 of file `vector.tcc`.

**5.1091.3.46** `resize()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
    size_type __new_size ) [inline]
```

Resizes the vector to the specified number of elements.

## Parameters

<code>__new_size</code>	Number of elements the vector should contain.
-------------------------	---

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 824 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`.

5.1091.3.47 `resize()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::resize (
    size_type __new_size,
    const value_type & __x ) [inline]
```

Resizes the vector to the specified number of elements.

## Parameters

<code>__new_size</code>	Number of elements the vector should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 844 of file `stl_vector.h`.

5.1091.3.48 `shrink_to_fit()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::shrink_to_fit ( ) [inline]
```

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 876 of file `stl_vector.h`.



### 5.1091.3.49 size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
size_type std::vector< _Tp, _Alloc >::size ( ) const [inline], [noexcept]
```

Returns the number of elements in the vector.

Definition at line 805 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::M_range_check()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution< _IntType >::max()`, `std::operator==()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize()`, and `std::regex_traits< _CharType >::transform_primary()`.

### 5.1091.3.50 swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::vector< _Tp, _Alloc >::swap (
    vector< _Tp, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another vector.

#### Parameters

<code>__x</code>	A vector of the same element and allocator types.
------------------	---

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1367 of file `stl_vector.h`.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 5.1092 std::vector< bool, \_Alloc > Class Template Reference

Inherits `std::_Bvector_base< _Alloc >`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**

## Public Member Functions

- **vector** (`const allocator_type &__a`)
- **vector** (`size_type __n, const allocator_type &__a=allocator_type()`)
- **vector** (`size_type __n, const bool &__value, const allocator_type &__a=allocator_type()`)
- **vector** (`const vector &__x`)
- **vector** (`vector &&__x`)=default
- **vector** (`vector &&__x, const allocator_type &__a`) noexcept(`_Bit_alloc_traits::_S_always_equal()`)
- **vector** (`const vector &__x, const allocator_type &__a`)
- **vector** (`initializer_list< bool > __l, const allocator_type &__a=allocator_type()`)
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
**vector** (`_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type()`)
- void **assign** (`size_type __n, const bool &__x`)
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
void **assign** (`_InputIterator __first, _InputIterator __last`)
- void **assign** (`initializer_list< bool > __l`)
- reference **at** (`size_type __n`)
- const\_reference **at** (`size_type __n`) const
- reference **back** ()
- const\_reference **back** () const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- size\_type **capacity** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- `const_reverse_iterator` **crbegin** () const noexcept
- `const_reverse_iterator` **crend** () const noexcept
- void **data** () noexcept
- template<typename... `_Args`>  
iterator **emplace** (`const_iterator __pos, _Args &&... __args`)
- template<typename... `_Args`>  
void **emplace\_back** (`_Args &&... __args`)
- bool **empty** () const noexcept
- iterator **end** () noexcept

- `const_iterator end ()` `const noexcept`
- `iterator erase (const_iterator __position)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `void flip ()` `noexcept`
- `reference front ()`
- `const_reference front ()` `const`
- `allocator_type get_allocator ()` `const`
- `iterator insert (const_iterator __position, const bool &__x=bool())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `iterator insert (const_iterator __position, size_type __n, const bool &__x)`
- `iterator insert (const_iterator __p, initializer\_list< bool > __l)`
- `size_type max_size ()` `const noexcept`
- `vector & operator= (const vector &__x)`
- `vector & operator= (vector &&__x)` `noexcept(_Bit_alloc_traits::_S_nothrow_move())`
- `vector & operator= (initializer\_list< bool > __l)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n)` `const`
- `void pop_back ()`
- `void push_back (bool __x)`
- `reverse\_iterator rbegin ()` `noexcept`
- `const\_reverse\_iterator rbegin ()` `const noexcept`
- `reverse\_iterator rend ()` `noexcept`
- `const\_reverse\_iterator rend ()` `const noexcept`
- `void reserve (size_type __n)`
- `void resize (size_type __new_size, bool __x=bool())`
- `void shrink_to_fit ()`
- `size_type size ()` `const noexcept`
- `void swap (vector &__x)` `noexcept`

#### Static Public Member Functions

- `static void swap (reference __x, reference __y)` `noexcept`

#### Protected Types

- `typedef \_\_gnu\_cxx::\_\_alloc\_traits< _Alloc >::template rebind< _Bit_type >::other \_Bit\_alloc\_type`

#### Protected Member Functions

- `_Bit_pointer _M_allocate (size_t __n)`
- `template<typename _InputIterator >`  
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `size_type _M_check_len (size_type __n, const char *__s)` `const`
- `iterator _M_copy_aligned (const_iterator __first, const_iterator __last, iterator __result)`
- `void _M_deallocate ()`

- iterator **\_M\_erase** (iterator \_\_pos)
- iterator **\_M\_erase** (iterator \_\_first, iterator \_\_last)
- void **\_M\_erase\_at\_end** (iterator \_\_pos)
- void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- `_Bit_alloc_type & _M_get_Bit_allocator () noexcept`
- `const _Bit_alloc_type & _M_get_Bit_allocator () const noexcept`
- void **\_M\_initialize** (size\_type \_\_n)
- `template<typename _Integer >`  
void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)
- `template<typename _InputIterator >`  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `template<typename _InputIterator >`  
void **\_M\_initialize\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- `template<typename _ForwardIterator >`  
void **\_M\_initialize\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void **\_M\_initialize\_value** (bool \_\_x)
- void **\_M\_insert\_aux** (iterator \_\_position, bool \_\_x)
- `template<typename _Integer >`  
void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)
- `template<typename _InputIterator >`  
void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `template<typename _InputIterator >`  
void **\_M\_insert\_range** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- `template<typename _ForwardIterator >`  
void **\_M\_insert\_range** (iterator \_\_position, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void **\_M\_move\_data** (\_Bvector\_base && \_\_x) noexcept
- void **\_M\_range\_check** (size\_type \_\_n) const
- void **\_M\_reallocate** (size\_type \_\_n)
- bool **\_M\_shrink\_to\_fit** ()

#### Static Protected Member Functions

- static size\_t **\_S\_nword** (size\_t \_\_n)

#### Protected Attributes

- `_Bvector_impl` **\_M\_impl**

#### Friends

- struct **std::hash< vector >**

#### 5.1092.1 Detailed Description

```
template<typename _Alloc>
class std::vector< bool, _Alloc >
```

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

#### Template Parameters

<code>_Alloc</code>	Allocator type.
---------------------	-----------------

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

#### See also

`vector` for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

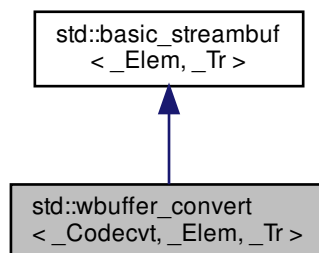
Definition at line 588 of file `stl_bvector.h`.

The documentation for this class was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

### 5.1093 `std::wbuffer_convert<_Codecvt, _Elem, _Tr >` Class Template Reference

Inheritance diagram for `std::wbuffer_convert<_Codecvt, _Elem, _Tr >`:



## Public Types

- typedef \_Codecvt::state\_type **state\_type**
- typedef \_Elem **char\_type**
- typedef \_Tr **traits\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef basic\_streambuf< char\_type, traits\_type > **\_\_streambuf\_type**

## Public Member Functions

- **wbuffer\_convert** (streambuf \* \_\_bytebuf=0, \_Codecvt \* \_\_pcvt=new \_Codecvt, state\_type \_\_state=state\_type())
- **wbuffer\_convert** (const **wbuffer\_convert** &)=delete
- **locale getloc** () const
- **streamsize in\_avail** ()
- **wbuffer\_convert** & **operator=** (const **wbuffer\_convert** &)=delete
- **locale pubimbue** (const **locale** & \_\_loc)
- **streambuf \* rdbuf** () const noexcept
- **streambuf \* rdbuf** (streambuf \* \_\_bytebuf) noexcept
- **int\_type sbumpc** ()
- **int\_type sgetc** ()
- **streamsize sgetn** (char\_type \* \_\_s, streamsize \_\_n)
- **int\_type snextc** ()
- **int\_type sputbackc** (char\_type \_\_c)
- **int\_type sputc** (char\_type \_\_c)
- **streamsize sputn** (const char\_type \* \_\_s, streamsize \_\_n)
- **state\_type state** () const noexcept
- **int\_type sungetc** ()
- **basic\_streambuf \* pubsetbuf** (char\_type \* \_\_s, streamsize \_\_n)
- **pos\_type pubseekoff** (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **pos\_type pubseekpos** (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **int pubsync** ()

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
  - void **\_\_safe\_pbump** (streamsize \_\_n)
  - void **gbump** (int \_\_n)
  - virtual void **imbue** (const locale &\_\_loc \_\_attribute\_\_((\_\_unused\_\_)))
  - **\_Wide\_streambuf::int\_type overflow** (typename **\_Wide\_streambuf::int\_type** \_\_out)
  - virtual **int\_type overflow** (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
  - virtual **int\_type pbackfail** (int\_type \_\_c \_\_attribute\_\_((\_\_unused\_\_))=traits\_type::eof())
  - void **pbump** (int \_\_n)
  - virtual **pos\_type seekoff** (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
  - virtual **pos\_type seekpos** (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
  - virtual **basic\_streambuf< char\_type, \_Tr > \* setbuf** (char\_type \*, streamsize)
  - void **setg** (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
  - void **setp** (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
  - virtual **streamsize showmanyc** ()
  - void **swap** (basic\_streambuf &\_\_sb)
  - int **sync** ()
  - virtual **int\_type uflow** ()
  - **\_Wide\_streambuf::int\_type underflow** ()
  - virtual **streamsize xsgetn** (char\_type \*\_\_s, streamsize \_\_n)
  - **streamsize xspun** (const typename **\_Wide\_streambuf::char\_type** \*\_\_s, streamsize \_\_n)
  - virtual **streamsize xspun** (const char\_type \*\_\_s, streamsize \_\_n)
- 
- **char\_type \* eback** () const
  - **char\_type \* gptr** () const
  - **char\_type \* egptr** () const
- 
- **char\_type \* pbase** () const
  - **char\_type \* pptr** () const
  - **char\_type \* ep\_ptr** () const

## Protected Attributes

- **locale \_M\_buf\_locale**
- **char\_type \* \_M\_in\_beg**
- **char\_type \* \_M\_in\_cur**
- **char\_type \* \_M\_in\_end**
- **char\_type \* \_M\_out\_beg**
- **char\_type \* \_M\_out\_cur**
- **char\_type \* \_M\_out\_end**

### 5.1093.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
```

Buffer conversions.

Definition at line 321 of file locale\_conv.h.

### 5.1093.2 Member Typedef Documentation

#### 5.1093.2.1 \_\_streambuf\_type

```
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _Elem , _Tr >::__streambuf_type
[inherited]
```

This is a non-standard type.

Definition at line 140 of file streambuf.

#### 5.1093.2.2 char\_type

```
typedef _Elem std::basic_streambuf< _Elem , _Tr >::char_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 131 of file streambuf.

#### 5.1093.2.3 int\_type

```
typedef traits_type::int_type std::basic_streambuf< _Elem , _Tr >::int_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 133 of file streambuf.



#### 5.1093.2.4 off\_type

```
typedef traits_type::off_type std::basic_streambuf< _Elem , _Tr >::off_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 135 of file streambuf.

#### 5.1093.2.5 pos\_type

```
typedef traits_type::pos_type std::basic_streambuf< _Elem , _Tr >::pos_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 134 of file streambuf.

#### 5.1093.2.6 traits\_type

```
typedef _Tr std::basic_streambuf< _Elem , _Tr >::traits_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 132 of file streambuf.

### 5.1093.3 Constructor & Destructor Documentation

#### 5.1093.3.1 wbuffer\_convert()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert (
    streambuf * __bytebuf = 0,
    _Codecvt * __pcvt = new _Codecvt,
    state_type __state = state_type() ) [inline], [explicit]
```

Default constructor.

##### Parameters

<code>__bytebuf</code>	The underlying byte stream buffer.
<code>__pcvt</code>	The facet to use for conversions.
<code>__state</code>	Initial conversion state.

Takes ownership of `__pcvt` and will delete it in the destructor.

Definition at line 337 of file `locale_conv.h`.

References `std::basic_streambuf<_Elem, _Tr>::setg()`, and `std::basic_streambuf<_Elem, _Tr>::setp()`.

#### 5.1093.4 Member Function Documentation

##### 5.1093.4.1 eback()

```
char_type* std::basic_streambuf<_Elem, _Tr>::eback ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

##### 5.1093.4.2 egptr()

```
char_type* std::basic_streambuf<_Elem, _Tr>::egptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

#### 5.1093.4.3 epptr()

```
char_type* std::basic_streambuf< _Elem , _Tr >::epptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 5.1093.4.4 gbump()

```
void std::basic_streambuf< _Elem , _Tr >::gbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the read position.

##### Parameters

<code>__n</code>	The delta by which to move.
------------------	-----------------------------

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

#### 5.1093.4.5 getloc()

```
locale std::basic_streambuf< _Elem , _Tr >::getloc ( ) const [inline], [inherited]
```

Locale access.

##### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

## 5.1093.4.6 gptr()

```
char_type* std::basic_streambuf<_Elem, _Tr>::gptr ( ) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

## 5.1093.4.7 imbue()

```
virtual void std::basic_streambuf<_Elem, _Tr>::imbue (
    const locale &__loc __attribute__((unused)) ) [inline], [protected], [virtual],
[inherited]
```

Changes translations.

## Parameters

<code>__loc</code>	A new locale.
--------------------	---------------

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Definition at line 583 of file streambuf.

## 5.1093.4.8 in\_avail()

```
streamsize std::basic_streambuf<_Elem, _Tr>::in_avail ( ) [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

**5.1093.4.9 overflow()**

```
virtual int_type std::basic_streambuf<_Elem, _Tr>::overflow (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

<code>__c</code>	An additional character to consume.
------------------	-------------------------------------

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 775 of file `streambuf`.

**5.1093.4.10 pbackfail()**

```
virtual int_type std::basic_streambuf<_Elem, _Tr>::pbackfail (
    int_type __c __attribute__((unused)) = traits_type::eof() ) [inline], [protected],
[virtual], [inherited]
```

Tries to back up the input sequence.

## Parameters

<code>_↵</code>	The character to be inserted back into the sequence.
<code>_C</code>	

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Definition at line 731 of file `streambuf`.

## 5.1093.4.11 pbase()

```
char_type* std::basic_streambuf< _Elem , _Tr >::pbase ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

## 5.1093.4.12 pbump()

```
void std::basic_streambuf< _Elem , _Tr >::pbump (
    int __n ) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

<code>_↵</code>	The delta by which to move.
<code>_n</code>	

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**5.1093.4.13 pptr()**

```
char_type* std::basic_streambuf<_Elem, _Tr>::pptr ( ) const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**5.1093.4.14 pubimbue()**

```
locale std::basic_streambuf<_Elem, _Tr>::pubimbue (
    const locale & __loc ) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

<code>__loc</code>	The new locale.
--------------------	-----------------

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

## 5.1093.4.15 pubseekoff()

```
pos_type std::basic_streambuf< _Elem , _Tr >::pubseekoff (
    off_type __off,
    ios_base::seekdir __way,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__off</code>	Offset.
<code>__way</code>	Value for <code>ios_base::seekdir</code> .
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

## 5.1093.4.16 pubseekpos()

```
pos_type std::basic_streambuf< _Elem , _Tr >::pubseekpos (
    pos_type __sp,
    ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline], [inherited]
```

Alters the stream position.

## Parameters

<code>__sp</code>	Position
<code>__mode</code>	Value for <code>ios_base::openmode</code> .

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

## 5.1093.4.17 pubsetbuf()

```
basic_streambuf* std::basic_streambuf< _Elem , _Tr >::pubsetbuf (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file streambuf.



**5.1093.4.18 pubsync()**

```
int std::basic_streambuf< _Elem , _Tr >::pubsync ( ) [inline], [inherited]
```

Calls virtual sync function.

Definition at line 278 of file streambuf.

**5.1093.4.19 sbumpc()**

```
int_type std::basic_streambuf< _Elem , _Tr >::sbumpc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file streambuf.

**5.1093.4.20 seekoff()**

```
virtual pos_type std::basic_streambuf< _Elem , _Tr >::seekoff (
    off_type ,
    ios_base::seekdir ,
    ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 609 of file streambuf.

#### 5.1093.4.21 `seekpos()`

```
virtual pos_type std::basic_streambuf<_Elem, _Tr>::seekpos (
    pos_type ,
    ios_base::openmode = ios_base::in | ios_base::out ) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 621 of file `streambuf`.

#### 5.1093.4.22 `setbuf()`

```
virtual basic_streambuf<char_type, _Tr>* std::basic_streambuf<_Elem, _Tr>::setbuf (
    char_type * ,
    streamsize ) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

##### Note

Base class version does nothing, returns `this`.

Definition at line 598 of file `streambuf`.

#### 5.1093.4.23 `setg()`

```
void std::basic_streambuf<_Elem, _Tr>::setg (
    char_type * __gbeg,
    char_type * __gnext,
    char_type * __gend ) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

<code>__gbeg</code>	A pointer.
<code>__gnext</code>	A pointer.
<code>__gend</code>	A pointer.

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file `streambuf`.

**5.1093.4.24 setp()**

```
void std::basic_streambuf<_Elem, _Tr>::setp (
    char_type * __pbeg,
    char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

<code>__pbeg</code>	A pointer.
<code>__pend</code>	A pointer.

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file `streambuf`.

**5.1093.4.25 sgetc()**

```
int_type std::basic_streambuf<_Elem, _Tr>::sgetc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

## 5.1093.4.26 sgetn()

```
streamsize std::basic_streambuf< _Elem , _Tr >::sgetn (
    char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for xsgetn.

## Parameters

<code>__s</code>	A buffer area.
<code>__n</code>	A count.

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

## 5.1093.4.27 showmanyc()

```
virtual streamsize std::basic_streambuf< _Elem , _Tr >::showmanyc ( ) [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1*

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Definition at line 656 of file `streambuf`.

**5.1093.4.28** `snextc()`

```
int_type std::basic_streambuf< _Elem , _Tr >::snextc ( ) [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**5.1093.4.29** `sputbackc()`

```
int_type std::basic_streambuf< _Elem , _Tr >::sputbackc (
    char_type __c ) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

<code>__c</code>	The character to push back.
------------------	-----------------------------

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**5.1093.4.30** `sputc()`

```
int_type std::basic_streambuf< _Elem , _Tr >::sputc (
    char_type __c ) [inline], [inherited]
```

Entry point for all single-character output functions.

## Parameters

<code>__c</code>	A character to output.
------------------	------------------------

## Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

## 5.1093.4.31 sputn()

```
streamsize std::basic_streambuf< _Elem , _Tr >::sputn (
    const char_type * __s,
    streamsize __n ) [inline], [inherited]
```

Entry point for all single-character output functions.

## Parameters

<code>__s</code>	A buffer read area.
<code>__n</code>	A count.

One of two public output functions.

Returns `xputn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

## 5.1093.4.32 state()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
state_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::state ( ) const [inline], [noexcept]
```

The conversion state following the last conversion.

Definition at line 373 of file `locale_conv.h`.

### 5.1093.4.33 `sungetc()`

```
int_type std::basic_streambuf< _Elem , _Tr >::sungetc ( ) [inline], [inherited]
```

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

### 5.1093.4.34 `sync()`

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
int std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync (
    void ) [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _Elem, _Tr >`.

Definition at line 377 of file `locale_conv.h`.

References `std::basic_streambuf< _CharT, _Traits >::pubsync()`.

#### 5.1093.4.35 uflow()

```
virtual int_type std::basic_streambuf< _Elem , _Tr >::uflow ( ) [inline], [protected], [virtual],  
[inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 707 of file `streambuf`.

#### 5.1093.4.36 underflow()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>  
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow ( ) [inline],  
[protected], [virtual]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _Elem, _Tr >`.

Definition at line 391 of file `locale_conv.h`.

References `std::basic_streambuf< _Elem, _Tr >::egptr()`, and `std::basic_streambuf< _Elem, _Tr >::gptr()`.

#### 5.1093.4.37 xsgetn()

```
streamsize std::basic_streambuf< _Elem , _Tr >::xsgetn (   
    char_type * __s,  
    streamsize __n ) [protected], [virtual], [inherited]
```

Multiple character extraction.



**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Definition at line 46 of file `streambuf.tcc`.

**5.1093.4.38 xspn()**

```
streamsize std::basic_streambuf<_Elem, _Tr>::xspn (
    const char_type * __s,
    streamsize __n ) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

<code>__s</code>	A buffer area.
<code>__n</code>	Maximum number of characters to write.

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Definition at line 80 of file `streambuf.tcc`.

**5.1093.5 Member Data Documentation**

#### 5.1093.5.1 \_M\_buf\_locale

`locale std::basic_streambuf<_Elem, _Tr>::_M_buf_locale` [protected], [inherited]

Current locale setting.

Definition at line 199 of file streambuf.

#### 5.1093.5.2 \_M\_in\_beg

`char_type* std::basic_streambuf<_Elem, _Tr>::_M_in_beg` [protected], [inherited]

Start of get area.

Definition at line 191 of file streambuf.

#### 5.1093.5.3 \_M\_in\_cur

`char_type* std::basic_streambuf<_Elem, _Tr>::_M_in_cur` [protected], [inherited]

Current read area.

Definition at line 192 of file streambuf.

#### 5.1093.5.4 \_M\_in\_end

`char_type* std::basic_streambuf<_Elem, _Tr>::_M_in_end` [protected], [inherited]

End of get area.

Definition at line 193 of file streambuf.

#### 5.1093.5.5 \_M\_out\_beg

`char_type* std::basic_streambuf<_Elem, _Tr>::_M_out_beg` [protected], [inherited]

Start of put area.

Definition at line 194 of file streambuf.

5.1093.5.6 `_M_out_cur`

```
char_type* std::basic_streambuf< _Elem , _Tr >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file `streambuf`.

5.1093.5.7 `_M_out_end`

```
char_type* std::basic_streambuf< _Elem , _Tr >::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file `streambuf`.

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

5.1094 `std::weak_ptr< _Tp >` Class Template Reference

Inherits `std::__weak_ptr< _Tp, _Lp >`.

## Public Types

- using **element\_type** = typename [remove\\_extent](#)< \_Tp >::type

## Public Member Functions

- `template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>`  
**weak\_ptr** (const [shared\\_ptr](#)< \_Yp > &\_\_r) noexcept
- **weak\_ptr** (const [weak\\_ptr](#) &) noexcept=default
- `template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>`  
**weak\_ptr** (const [weak\\_ptr](#)< \_Yp > &\_\_r) noexcept
- **weak\_ptr** ([weak\\_ptr](#) &&) noexcept=default
- `template<typename _Yp, typename = _Constructible<weak_ptr<_Yp>>>`  
**weak\_ptr** ([weak\\_ptr](#)< \_Yp > &&\_\_r) noexcept
- `bool expired ()` const noexcept
- `shared\_ptr< _Tp > lock ()` const noexcept
- `weak\_ptr & operator= (const weak\_ptr &__r)` noexcept=default
- `template<typename _Yp >`  
`_Assignable< const weak\_ptr< _Yp > & > operator= (const weak\_ptr< _Yp > &__r)` noexcept
- `template<typename _Yp >`  
`_Assignable< const shared\_ptr< _Yp > & > operator= (const shared\_ptr< _Yp > &__r)` noexcept
- `weak\_ptr & operator= (weak\_ptr &&__r)` noexcept=default
- `template<typename _Yp >`  
`_Assignable< weak\_ptr< _Yp > > operator= (weak\_ptr< _Yp > &&__r)` noexcept
- `template<typename _Tp1 >`  
`bool owner_before (const __shared_ptr< _Tp1, _Lp > &__rhs)` const noexcept
- `template<typename _Tp1 >`  
`bool owner_before (const __weak_ptr< _Tp1, _Lp > &__rhs)` const noexcept
- `void reset ()` noexcept
- `void swap (__weak_ptr &__s)` noexcept
- `long use_count ()` const noexcept

## 5.1094.1 Detailed Description

```
template<typename _Tp>
class std::weak_ptr<_Tp>
```

A smart pointer with weak semantics.

With forwarding constructors and assignment operators.

Definition at line 535 of file `bits/shared_ptr.h`.

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

5.1095 `std::weibull_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **`weibull_distribution`** (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- **`weibull_distribution`** (const [param\\_type](#) &\_\_p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const`  
`param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` (`result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const`  
`param\_type &__p`)
- `_RealType a` () const
- `_RealType b` () const
- `result\_type max` () const
- `result\_type min` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator()` (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `param\_type param` () const
- `void param` (const [param\\_type](#) &\_\_param)
- `void reset` ()

## Friends

- bool `operator==(const weibull_distribution &__d1, const weibull_distribution &__d2)`

### 5.1095.1 Detailed Description

```
template<typename _RealType = double>
class std::weibull_distribution< _RealType >
```

A weibull\_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$$

Definition at line 4758 of file random.h.

### 5.1095.2 Member Typedef Documentation

#### 5.1095.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::weibull_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 4761 of file random.h.

### 5.1095.3 Member Function Documentation

#### 5.1095.3.1 a()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::a ( ) const [inline]
```

Return the *a* parameter of the distribution.

Definition at line 4821 of file random.h.

Referenced by `std::operator<<()`.

### 5.1095.3.2 `b()`

```
template<typename _RealType = double>
_RealType std::weibull_distribution<_RealType>::b ( ) const [inline]
```

Return the  $b$  parameter of the distribution.

Definition at line 4828 of file `random.h`.

Referenced by `std::operator<<()`.

### 5.1095.3.3 `max()`

```
template<typename _RealType = double>
result_type std::weibull_distribution<_RealType>::max ( ) const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4857 of file `random.h`.

References `std::numeric_limits<_Tp>::max()`.

### 5.1095.3.4 `min()`

```
template<typename _RealType = double>
result_type std::weibull_distribution<_RealType>::min ( ) const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4850 of file `random.h`.

### 5.1095.3.5 `operator()()`

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::weibull_distribution<_RealType>::operator() (
    _UniformRandomNumberGenerator & __urng ) [inline]
```

Generating functions.

Definition at line 4865 of file `random.h`.

**5.1095.3.6** `param()` [1/2]

```
template<typename _RealType = double>
param_type std::weibull_distribution< _RealType >::param ( ) const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4835 of file random.h.

Referenced by `std::operator>>()`.

**5.1095.3.7** `param()` [2/2]

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::param (
    const param_type & __param ) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4843 of file random.h.

**5.1095.3.8** `reset()`

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 4814 of file random.h.

**5.1095.4 Friends And Related Function Documentation****5.1095.4.1** `operator==`

```
template<typename _RealType = double>
bool operator== (
    const weibull_distribution< _RealType > & __d1,
    const weibull_distribution< _RealType > & __d2 ) [friend]
```

Return true if two Weibull distributions have the same parameters.

Definition at line 4900 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 5.1096 `std::weibull_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [weibull\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_a=\_RealType(1), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.1096.1 Detailed Description

```
template<typename _RealType = double>
struct std::weibull_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 4768 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.1097 `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc>` Class Template Reference

### Public Types

- typedef [basic\\_string](#)<char, [char\\_traits](#)<char>, \_Byte\_alloc> **byte\_string**
- typedef [wstring::traits\\_type::int\\_type](#) **int\_type**
- typedef \_Codecvt::state\_type **state\_type**
- typedef [basic\\_string](#)<\_Elem, [char\\_traits](#)<\_Elem>, \_Wide\_alloc> **wide\_string**



## Public Member Functions

- [wstring\\_convert](#) ([\\_Codecvt](#) \* \_\_pcvt=new [\\_Codecvt](#)())
- [wstring\\_convert](#) ([\\_Codecvt](#) \* \_\_pcvt, [state\\_type](#) \_\_state)
- [wstring\\_convert](#) (const [byte\\_string](#) & \_\_byte\_err, const [wide\\_string](#) & \_\_wide\_err=[wide\\_string](#)())
- [wstring\\_convert](#) (const [wstring\\_convert](#) &)=delete
- [size\\_t](#) [converted](#) () const noexcept
- [wstring\\_convert](#) & **operator=** (const [wstring\\_convert](#) &)=delete
- [state\\_type](#) [state](#) () const

- [wide\\_string from\\_bytes](#) (char \_\_byte)
- [wide\\_string from\\_bytes](#) (const char \* \_\_ptr)
- [wide\\_string from\\_bytes](#) (const [byte\\_string](#) & \_\_str)
- [wide\\_string from\\_bytes](#) (const char \* \_\_first, const char \* \_\_last)

- [byte\\_string to\\_bytes](#) ([\\_Elem](#) \_\_wchar)
- [byte\\_string to\\_bytes](#) (const [\\_Elem](#) \* \_\_ptr)
- [byte\\_string to\\_bytes](#) (const [wide\\_string](#) & \_\_wstr)
- [byte\\_string to\\_bytes](#) (const [\\_Elem](#) \* \_\_first, const [\\_Elem](#) \* \_\_last)

## 5.1097.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>, typename _Byte_alloc =
allocator<char>>>
class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >
```

String conversions.

Definition at line 169 of file locale\_conv.h.

## 5.1097.2 Constructor &amp; Destructor Documentation

5.1097.2.1 [wstring\\_convert](#)() [1/3]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
    _Codecvt * __pcvt = new _Codecvt() ) [inline], [explicit]
```

Default constructor.

**Parameters**

<code>__pcvt</code>	The facet to use for conversions.
---------------------	-----------------------------------

Takes ownership of `__pcvt` and will delete it in the destructor.

Definition at line 184 of file `locale_conv.h`.

**5.1097.2.2 wstring\_convert()** [2/3]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
    _Codecvt * __pcvt,
    state_type __state ) [inline]
```

Construct with an initial conversion state.

**Parameters**

<code>__pcvt</code>	The facet to use for conversions.
<code>__state</code>	Initial conversion state.

Takes ownership of `__pcvt` and will delete it in the destructor. The object's conversion state will persist between conversions.

Definition at line 198 of file `locale_conv.h`.

**5.1097.2.3 wstring\_convert()** [3/3]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
    const byte_string & __byte_err,
    const wide_string & __wide_err = wide_string() ) [inline], [explicit]
```

Construct with error strings.

**Parameters**

<code>__byte_err</code>	A string to return on failed conversions.
<code>__wide_err</code>	A wide string to return on failed conversions.

Definition at line 211 of file `locale_conv.h`.

### 5.1097.3 Member Function Documentation

#### 5.1097.3.1 converted()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
size_t std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::converted ( ) const
[inline], [noexcept]
```

The number of elements successfully converted in the last conversion.

Definition at line 301 of file locale\_conv.h.

#### 5.1097.3.2 from\_bytes() [1/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    char __byte ) [inline]
```

Convert from bytes.

Definition at line 230 of file locale\_conv.h.

Referenced by std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::from\_bytes().

#### 5.1097.3.3 from\_bytes() [2/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    const char * __ptr ) [inline]
```

Convert from bytes.

Definition at line 237 of file locale\_conv.h.

References std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::from\_bytes().

## 5.1097.3.4 from\_bytes() [3/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    const byte_string & __str ) [inline]
```

Convert from bytes.

Definition at line 241 of file locale\_conv.h.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::data(), std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::from\_bytes(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::size().

## 5.1097.3.5 from\_bytes() [4/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
    const char * __first,
    const char * __last ) [inline]
```

Convert from bytes.

Definition at line 248 of file locale\_conv.h.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::get\_allocator().

## 5.1097.3.6 state()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
state_type std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::state ( ) const
[inline]
```

The final conversion state of the last conversion.

Definition at line 304 of file locale\_conv.h.

## 5.1097.3.7 to\_bytes() [1/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    _Elem __wchar ) [inline]
```

Convert to bytes.

Definition at line 264 of file locale\_conv.h.

Referenced by std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::to\_bytes().

**5.1097.3.8 to\_bytes()** [2/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    const _Elem * __ptr ) [inline]
```

Convert to bytes.

Definition at line 271 of file locale\_conv.h.

References `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes()`.

**5.1097.3.9 to\_bytes()** [3/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    const wide_string & __wstr ) [inline]
```

Convert to bytes.

Definition at line 277 of file locale\_conv.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::size()`, and `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes()`.

**5.1097.3.10 to\_bytes()** [4/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
    const _Elem * __first,
    const _Elem * __last ) [inline]
```

Convert to bytes.

Definition at line 284 of file locale\_conv.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`.

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

## 6 File Documentation

### 6.1 algo.h File Reference

#### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber< \\_MustBeInt >](#)

#### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

#### Functions

- template<typename \_RAIter >  
\_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, random\_access\_iterator\_tag)
- template<typename \_FIterator, typename \_IteratorTag >  
\_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_IteratorTag)
- template<typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag >  
\_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_BinaryPredicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_BinaryPredicate >  
\_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_BinaryPredicate \_\_pred, random\_access\_iterator\_tag)
- template<typename \_RAIter, typename \_Predicate >  
iterator\_traits< \_RAIter >::difference\_type **std::\_\_parallel::\_\_count\_if\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag)
- template<typename \_Iter, typename \_Predicate, typename \_IteratorTag >  
iterator\_traits< \_Iter >::difference\_type **std::\_\_parallel::\_\_count\_if\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Predicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_Tp >  
iterator\_traits< \_RAIter >::difference\_type **std::\_\_parallel::\_\_count\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, const \_Tp & \_\_value, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag)
- template<typename \_Iter, typename \_Tp, typename \_IteratorTag >  
iterator\_traits< \_Iter >::difference\_type **std::\_\_parallel::\_\_count\_switch** (\_Iter \_\_begin, \_Iter \_\_end, const \_Tp & \_\_value, \_IteratorTag)
- template<typename \_Iter, typename \_FIterator, typename \_IteratorTag1, typename \_IteratorTag2 >  
\_Iter **std::\_\_parallel::\_\_find\_first\_of\_switch** (\_Iter \_\_begin1, \_Iter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, \_IteratorTag1, \_IteratorTag2)
- template<typename \_RAIter, typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag >  
\_RAIter **std::\_\_parallel::\_\_find\_first\_of\_switch** (\_RAIter \_\_begin1, \_RAIter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, \_BinaryPredicate \_\_comp, random\_access\_iterator\_tag, \_IteratorTag)
- template<typename \_Iter, typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag1, typename \_IteratorTag2 >  
\_Iter **std::\_\_parallel::\_\_find\_first\_of\_switch** (\_Iter \_\_begin1, \_Iter \_\_end1, \_FIterator \_\_begin2, \_FIterator \_\_end2, \_BinaryPredicate \_\_comp, \_IteratorTag1, \_IteratorTag2)
- template<typename \_Iter, typename \_Predicate, typename \_IteratorTag >  
\_Iter **std::\_\_parallel::\_\_find\_if\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Predicate \_\_pred, \_IteratorTag)

- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_↵`  
`access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter std::parallel::find_switch ( _Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch ( _RAIter __begin, _RAIter __end, const _Tp &__val, random_access_↵`  
`iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`  
`_Function std::parallel::for_each_switch ( _Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::for_each_switch ( _RAIter __begin, _RAIter __end, _Function __f, random_↵`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`  
`_OutputIterator std::parallel::generate_n_switch ( _OutputIterator __begin, _Size __n, _Generator __gen,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch ( _RAIter __begin, _Size __n, _Generator __gen, random_↵`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`  
`void std::parallel::generate_switch ( _FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch ( _RAIter __begin, _RAIter __end, _Generator __gen, random_↵`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::max_element_switch ( _FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::max_element_switch ( _RAIter __begin, _RAIter __end, _Compare __comp,`  
`random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _↵`  
`IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::merge_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 ↵`  
`__end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::parallel::merge_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 ↵`  
`__end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::min_element_switch ( _FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::min_element_switch ( _RAIter __begin, _RAIter __end, _Compare __comp,`  
`random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator std::parallel::partition_switch ( _FIterator __begin, _FIterator __end, _Predicate __pred, ↵`  
`IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::partition_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_↵`  
`access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void std::parallel::replace_if_switch ( _FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp`  
`&__new_value, _IteratorTag)`

- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &↵`  
`__new_value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void std::parallel::replace_switch ( _FIterator __begin, _FIterator __end, const _Tp &__old_value, const`  
`_Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::parallel::replace_switch ( _RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp`  
`&__new_value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter std::parallel::search_n_switch ( _RAIter __begin, _RAIter __end, _Integer __count, const _Tp &↵`  
`__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator std::parallel::search_n_switch ( _FIterator __begin, _FIterator __end, _Integer __count, const`  
`_Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch ( _FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2,`  
`_FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 std::parallel::search_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch ( _FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2,`  
`_FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename`  
`_IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_difference_switch ( _IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↵`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random↵`  
`__access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename`  
`_IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_intersection_switch ( _IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_intersection_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RA↵`  
`Iter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,`  
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename`  
`_IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_symmetric_difference_switch ( _IIter1 __begin1, _IIter1 __end1, _IIter2`  
`__begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, ↵`  
`Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1,`  
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator ↵`  
`tag, random_access_iterator_tag, random_access_iterator_tag)`



- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`  
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`  
`RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter>::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter>::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter>::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits<_Iter>::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits<_Iter>::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag, BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag, BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`

- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism ↵`  
`__parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`

- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _RAlter >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

- [illegible]

- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`\_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, \_OutputIterator __result, \_UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`

### 6.1.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header.

The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 6.2 `algbase.h` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)



## Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool std::parallel::equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool std::parallel::lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`



- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`

### 6.2.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 6.3 algorithm File Reference

### Macros

- `#define _GLIBCXX_ALGORITHM`

### 6.3.1 Detailed Description

This is a Standard C++ Library header.

## 6.4 algorithm File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_ALGORITHM`

## Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > \_\_gnu\_cxx::\_\_copy\_n ( _InputIterator __first, _Size __count, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`  
`pair< _RAIterator, _OutputIterator > \_\_gnu\_cxx::\_\_copy\_n ( _RAIterator __first, _Size __count, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int \_\_gnu\_cxx::\_\_lexicographical\_compare\_3way ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `int \_\_gnu\_cxx::\_\_lexicographical\_compare\_3way (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)`
- `int \_\_gnu\_cxx::\_\_lexicographical\_compare\_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `template<typename _Tp >`  
`const _Tp & \_\_gnu\_cxx::\_\_median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & \_\_gnu\_cxx::\_\_median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`  
`_RandomAccessIterator \_\_gnu\_cxx::\_\_random\_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator \_\_gnu\_cxx::\_\_random\_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > \_\_gnu\_cxx::copy\_n ( _InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`  
`void \_\_gnu\_cxx::count ( _InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`  
`void \_\_gnu\_cxx::count\_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int \_\_gnu\_cxx::lexicographical\_compare\_3way ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator \_\_gnu\_cxx::random\_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator \_\_gnu\_cxx::random\_sample ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator \_\_gnu\_cxx::random\_sample\_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator \_\_gnu\_cxx::random\_sample\_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)`

#### 6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

### 6.5 algorithm File Reference

#### Macros

- `#define _PARALLEL_ALGORITHM`

#### 6.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 6.6 algorithm File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define __cpp_lib_experimental_sample`
- `#define _GLIBCXX_EXPERIMENTAL_ALGORITHM`

#### Functions

- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>  
<_SampleIterator std::experimental::fundamentals\_v2::sample (_PopulationIterator __first, _PopulationIterator __↔  
_last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>  
_SampleIterator std::experimental::fundamentals_v2::sample (_PopulationIterator __first, _PopulationIterator  
_last, _SampleIterator __out, _Distance __n)`
- `template<typename _ForwardIterator, typename _Searcher>  
_ForwardIterator std::experimental::fundamentals_v2::search (_ForwardIterator __first, _ForwardIterator __↔  
_last, const _Searcher &__searcher)`
- `template<typename _RandomAccessIterator>  
void std::experimental::fundamentals_v2::shuffle (_RandomAccessIterator __first, _RandomAccessIterator  
_last)`

#### 6.6.1 Detailed Description

This is a TS C++ Library header.

## 6.6.2 Function Documentation

## 6.6.2.1 sample()

```
template<typename _PopulationIterator , typename _SampleIterator , typename _Distance , typename
_UniformRandomNumberGenerator >
_SampleIterator std::experimental::fundamentals_v2::sample (
    _PopulationIterator __first,
    _PopulationIterator __last,
    _SampleIterator __out,
    _Distance __n,
    _UniformRandomNumberGenerator && __g )
```

Take a random sample from a population.

Definition at line 60 of file experimental/algorithm.

## 6.7 algorithmfwd.h File Reference

## Namespaces

- [std](#)

## Functions

- `template<typename _Filter >`  
`_Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter , typename _BinaryPredicate >`  
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter , typename _Predicate >`  
`bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter , typename _Predicate >`  
`bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter , typename _Tp >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter , typename _Tp , typename _Compare >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Iter , typename _OIter >`  
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1 , typename _BIter2 >`  
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter , typename _OIter , typename _Predicate >`  
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter , typename _Size , typename _OIter >`  
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter , typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::count_if ( _Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::equal ( _Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal ( _Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > std::equal_range ( _Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > std::equal_range ( _Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`void std::fill ( _Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter std::fill_n ( _OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::find ( _Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_end ( _Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_end ( _Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_first_of ( _Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_first_of ( _Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if ( _Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if_not ( _Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`  
`_Funct std::for_each ( _Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`  
`void std::generate ( _Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::generate_n ( _OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::includes ( _Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::includes ( _Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`  
`void std::inplace_merge ( _BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void std::inplace_merge ( _BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`bool std::is_heap ( _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool std::is_heap ( _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`_RAIter std::is_heap_until ( _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::is_heap_until ( _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::is_partitioned ( _Iter, _Iter, _Predicate)`

- `template<typename _Filter1, typename _Filter2 >`  
`bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`bool std::is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _Filter >`  
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`  
`void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAlter >`  
`void std::make_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`  
`void std::make_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_GLIBCXX14_CONSTEXPR _Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_GLIBCXX14_CONSTEXPR _Filter std::min_element (_Filter, _Filter)`

- `template<typename _Filter, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_GLIBCXX14_CONSTEXPR pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _BIter >`  
`bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::none_of (_Iter, _Iter, _Predicate)`
- `template<typename _RAIter >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`  
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _Iter, typename _Olter1, typename _Olter2, typename _Predicate >`  
`pair< _Olter1, _Olter2 > std::partition_copy (_Iter, _Iter, _Olter1, _Olter2, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter >`  
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter >`  
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`void std::push_heap (_RAIter, _RAIter)`

- `template<typename _RAIter, typename _Compare >`  
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`  
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`_OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp >`  
`void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`_OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter std::replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _BIter >`  
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`  
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _Filter >`  
`_Filter std::_V2::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _OIter >`  
`_OIter std::rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`  
`_Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`_Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`



- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAIter, typename _UGenerator >`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter >`  
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`  
`_OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Filter >`  
`_Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`  
`_OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp >`  
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

### 6.7.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.8 `algorithmfwd.h` File Reference

### Namespaces

- `std`
- `std::__parallel`

## Functions

- `template<typename _Filter, typename _IterTag >`  
`_Filter std::parallel::adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_Filter std::parallel::adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter >`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_RAIter std::parallel::find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter std::parallel::find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter std::parallel::find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function std::parallel::for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`  
`void std::parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool std::__parallel::__lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::__parallel::__lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::__parallel::__max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::__parallel::__max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Olter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_Olter std::__parallel::__merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Olter, typename _Compare >`  
`_Olter std::__parallel::__merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::__parallel::__min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::__parallel::__min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::__parallel::__mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::__mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`  
`_Filter std::__parallel::__partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::__parallel::__partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void std::__parallel::__replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::__parallel::__replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`  
`void std::__parallel::__replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::__parallel::__replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter std::__parallel::__search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_Filter std::__parallel::__search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::__parallel::__search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 std::__parallel::__search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`  
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`  
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`

- `template<typename _RAIter, typename _RandomAccess_Olter, typename _Predicate >`  
`_RandomAccess_Olter std::__parallel::__unique_copy_switch (_RAIter, _RAIter, _RandomAccess_Olter, _↵`  
`Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`  
`value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`  
`value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`  
`value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`  
`pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`  
`pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`  
`pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism  
\_\_parallelism\_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`

- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _RAlter >`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter >`  
`void std::__parallel::nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter >`  
`void std::__parallel::partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _RAlter >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end)`



- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`



- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAlter >`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::sort (_RAlter __begin, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter >`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAlter __begin, _RAlter __end, _Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`

### 6.8.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 6.9 aligned\_buffer.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 6.9.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.10 alloc\_traits.h File Reference

### Classes

- struct [std::allocator\\_traits<\\_Alloc>](#)
- struct [std::allocator\\_traits<allocator<\\_Tp>>](#)

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_allocator_traits_is_always_equal`

### Typedefs

- `template<typename _Alloc, typename _Up>`  
using [std::\\_\\_alloc\\_rebind](#) = `typename __allocator_traits_base::template __rebind<_Alloc, _Up>::type`
- `template<typename _Alloc>`  
using [std::\\_RequireAllocator](#) = `typename enable_if<__is_allocator<_Alloc>::value, _Alloc>::type`

### Functions

- `template<typename _Alloc>`  
void [std::\\_\\_alloc\\_on\\_copy](#) (`_Alloc &__one`, `const _Alloc &__two`)
- `template<typename _Alloc>`  
`_Alloc` [std::\\_\\_alloc\\_on\\_copy](#) (`const _Alloc &__a`)
- `template<typename _Alloc>`  
void [std::\\_\\_alloc\\_on\\_move](#) (`_Alloc &__one`, `_Alloc &__two`)
- `template<typename _Alloc>`  
void [std::\\_\\_alloc\\_on\\_swap](#) (`_Alloc &__one`, `_Alloc &__two`)
- `template<typename _Alloc>`  
void [std::\\_\\_do\\_alloc\\_on\\_copy](#) (`_Alloc &__one`, `const _Alloc &__two`, `true_type`)
- `template<typename _Alloc>`  
void [std::\\_\\_do\\_alloc\\_on\\_copy](#) (`_Alloc &`, `const _Alloc &`, `false_type`)
- `template<typename _Alloc>`  
void [std::\\_\\_do\\_alloc\\_on\\_move](#) (`_Alloc &__one`, `_Alloc &__two`, `true_type`)
- `template<typename _Alloc>`  
void [std::\\_\\_do\\_alloc\\_on\\_move](#) (`_Alloc &`, `_Alloc &`, `false_type`)
- `template<typename _Alloc>`  
void [std::\\_\\_do\\_alloc\\_on\\_swap](#) (`_Alloc &__one`, `_Alloc &__two`, `true_type`)
- `template<typename _Alloc>`  
void [std::\\_\\_do\\_alloc\\_on\\_swap](#) (`_Alloc &`, `_Alloc &`, `false_type`)

### 6.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.11 `alloc_traits.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits<\\_Alloc, typename >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 6.11.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.12 `allocated_ptr.h` File Reference

### Classes

- struct [std::\\_\\_allocated\\_ptr<\\_Alloc >](#)

### Namespaces

- [std](#)

### Functions

- template<typename \_Alloc >  
[\\_\\_allocated\\_ptr<\\_Alloc >](#) [std::\\_\\_allocate\\_guarded](#) (\_Alloc &\_\_a)

### 6.12.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.13 allocator.h File Reference

### Classes

- class [std::allocator< \\_Tp >](#)
- class [std::allocator< void >](#)

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_allocator_is_always_equal`
- `#define __cpp_lib_incomplete_container_elements`

### Functions

- `template<typename _T1 , typename _T2 >`  
`bool std::operator!= (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _Tp >`  
`bool std::operator!= (const allocator< _Tp > &, const allocator< _Tp > &) noexcept`
- `template<typename _T1 , typename _T2 >`  
`bool std::operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _Tp >`  
`bool std::operator== (const allocator< _Tp > &, const allocator< _Tp > &) noexcept`

#### 6.13.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.14 any File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::any](#)
- class [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#)

### Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_experimental_any`
- `#define _GLIBCXX_EXPERIMENTAL_ANY`

## Functions

- `void std::experimental::fundamentals_v1::__throw_bad_any_cast ()`
- `template<typename _ValueType >  
_ValueType std::experimental::fundamentals_v1::any_cast (const any &__any)`
- `void std::experimental::fundamentals_v1::swap (any &__x, any &__y) noexcept`
  
- `template<typename _ValueType >  
_ValueType std::experimental::fundamentals_v1::any_cast (any &__any)`
- `template<typename _ValueType , typename enable_if<is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>  
_ValueType std::experimental::fundamentals_v1::any_cast (any &&__any)`
  
- `template<typename _ValueType >  
const _ValueType * std::experimental::fundamentals_v1::any_cast (const any * __any) noexcept`
- `template<typename _ValueType >  
_ValueType * std::experimental::fundamentals_v1::any_cast (any * __any) noexcept`

### 6.14.1 Detailed Description

This is a TS C++ Library header.

## 6.15 array File Reference

### Classes

- `struct std::array< _Tp, _Nm >`
- `struct std::tuple_element< _Int, _Tp >`
- `struct std::tuple_element< _Int, ::array< _Tp, _Nm > >`
- `struct std::tuple_size< _Tp >`
- `struct std::tuple_size<::array< _Tp, _Nm > >`

### Namespaces

- `std`

## Macros

- `#define _GLIBCXX_ARRAY`

## Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp & std::get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp && std::get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp & std::get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp && std::get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`enable_if< ::__array_traits< _Tp, _Nm >::is_swappable::value >::type std::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`
- `template<typename _Tp, std::size_t _Nm>`  
`enable_if< !::__array_traits< _Tp, _Nm >::is_swappable::value >::type std::swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`

## 6.15.1 Detailed Description

This is a Standard C++ Library header.

## 6.16 array File Reference

## Classes

- `struct std::tuple\_element< \_Int, std::\_\_debug::array< \_Tp, \_Nm > >`
- `struct std::tuple\_size< std::\_\_debug::array< \_Tp, \_Nm > >`

## Namespaces

- `std`
- `std::\_\_debug`

## Macros

- `#define _GLIBCXX_DEBUG_ARRAY`

## Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & std::__debug::get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && std::__debug::get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & std::__debug::get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp && std::__debug::get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>  
bool std::__debug::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::__debug::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::__debug::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::__debug::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::__debug::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::__debug::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, size_t _Nm>  
enable_if< !::__array_traits< _Tp, _Nm >::is_swappable::value >::type std::__debug::swap (array< _Tp, ↵  
_Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, std::size_t _Nm>  
void std::__debug::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(↵  
one.swap(__two)))`

### 6.16.1 Detailed Description

This is a Standard C++ Library header.

## 6.17 array File Reference

### Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_experimental_make_array`
- `#define _GLIBCXX_EXPERIMENTAL_ARRAY`

## Functions

- `template<typename _Tp, size_t _Nm, size_t... _Idx>  
constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (_Tp(&__a)[_Nm], index_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>  
constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::fundamentals_v2::make_array (_Types &&... __t)`
- `template<typename _Tp, size_t _Nm>  
constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)`

## 6.17.1 Detailed Description

This is a TS C++ Library header.

## 6.18 array\_allocator.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::array\\_allocator< \\_Tp, \\_Array >](#)
- class [\\_\\_gnu\\_cxx::array\\_allocator\\_base< \\_Tp >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp, typename _Array >  
bool __gnu_cxx::operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp, typename _Array >  
bool __gnu_cxx::operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

## 6.18.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.19 assertions.h File Reference

## Macros

- `#define __glibcxx_requires_non_empty_range( _First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_subscript( _N)`
- `#define _GLIBCXX_DEBUG_ASSERT( _Condition)`
- `#define _GLIBCXX_DEBUG_ONLY( _Statement)`
- `#define _GLIBCXX_DEBUG_PEDASSERT( _Condition)`



### 6.19.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.20 `assoc_container.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::basic\\_branch](#)< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >
- class [\\_\\_gnu\\_pbds::basic\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc >
- class [\\_\\_gnu\\_pbds::cc\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class [\\_\\_gnu\\_pbds::gp\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class [\\_\\_gnu\\_pbds::list\\_update](#)< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >
- class [\\_\\_gnu\\_pbds::tree](#)< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie](#)< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`
- `#define PB_DS_LU_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

### 6.20.1 Detailed Description

Contains associative containers.

## 6.21 atomic File Reference

### Classes

- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic< \\_Tp \\* >](#)
- struct [std::atomic< bool >](#)
- struct [std::atomic< char >](#)
- struct [std::atomic< char16\\_t >](#)
- struct [std::atomic< char32\\_t >](#)
- struct [std::atomic< int >](#)
- struct [std::atomic< long >](#)
- struct [std::atomic< long long >](#)
- struct [std::atomic< short >](#)
- struct [std::atomic< signed char >](#)
- struct [std::atomic< unsigned char >](#)
- struct [std::atomic< unsigned int >](#)
- struct [std::atomic< unsigned long >](#)
- struct [std::atomic< unsigned long long >](#)
- struct [std::atomic< unsigned short >](#)
- struct [std::atomic< wchar\\_t >](#)

### Namespaces

- [std](#)

### Macros

- `#define \_GLIBCXX\_ATOMIC`

### Typedefs

- typedef atomic< bool > [std::atomic\\_bool](#)
- typedef atomic< char > [std::atomic\\_char](#)
- typedef atomic< char16\_t > [std::atomic\\_char16\\_t](#)
- typedef atomic< char32\_t > [std::atomic\\_char32\\_t](#)
- typedef atomic< int > [std::atomic\\_int](#)
- typedef atomic< int16\_t > [std::atomic\\_int16\\_t](#)
- typedef atomic< int32\_t > [std::atomic\\_int32\\_t](#)
- typedef atomic< int64\_t > [std::atomic\\_int64\\_t](#)
- typedef atomic< int8\_t > [std::atomic\\_int8\\_t](#)
- typedef atomic< int\_fast16\_t > [std::atomic\\_int\\_fast16\\_t](#)
- typedef atomic< int\_fast32\_t > [std::atomic\\_int\\_fast32\\_t](#)
- typedef atomic< int\_fast64\_t > [std::atomic\\_int\\_fast64\\_t](#)
- typedef atomic< int\_fast8\_t > [std::atomic\\_int\\_fast8\\_t](#)
- typedef atomic< int\_least16\_t > [std::atomic\\_int\\_least16\\_t](#)
- typedef atomic< int\_least32\_t > [std::atomic\\_int\\_least32\\_t](#)

- `typedef atomic< int_least64_t > std::atomic_int_least64_t`
- `typedef atomic< int_least8_t > std::atomic_int_least8_t`
- `typedef atomic< intmax_t > std::atomic_intmax_t`
- `typedef atomic< intptr_t > std::atomic_intptr_t`
- `typedef atomic< long long > std::atomic_llong`
- `typedef atomic< long > std::atomic_long`
- `typedef atomic< ptrdiff_t > std::atomic_ptrdiff_t`
- `typedef atomic< signed char > std::atomic_schar`
- `typedef atomic< short > std::atomic_short`
- `typedef atomic< size_t > std::atomic_size_t`
- `typedef atomic< unsigned char > std::atomic_uchar`
- `typedef atomic< unsigned int > std::atomic_uint`
- `typedef atomic< uint16_t > std::atomic_uint16_t`
- `typedef atomic< uint32_t > std::atomic_uint32_t`
- `typedef atomic< uint64_t > std::atomic_uint64_t`
- `typedef atomic< uint8_t > std::atomic_uint8_t`
- `typedef atomic< uint_fast16_t > std::atomic_uint_fast16_t`
- `typedef atomic< uint_fast32_t > std::atomic_uint_fast32_t`
- `typedef atomic< uint_fast64_t > std::atomic_uint_fast64_t`
- `typedef atomic< uint_fast8_t > std::atomic_uint_fast8_t`
- `typedef atomic< uint_least16_t > std::atomic_uint_least16_t`
- `typedef atomic< uint_least32_t > std::atomic_uint_least32_t`
- `typedef atomic< uint_least64_t > std::atomic_uint_least64_t`
- `typedef atomic< uint_least8_t > std::atomic_uint_least8_t`
- `typedef atomic< uintmax_t > std::atomic_uintmax_t`
- `typedef atomic< uintptr_t > std::atomic_uintptr_t`
- `typedef atomic< unsigned long long > std::atomic_ullong`
- `typedef atomic< unsigned long > std::atomic_ulong`
- `typedef atomic< unsigned short > std::atomic_ushort`
- `typedef atomic< wchar_t > std::atomic_wchar_t`

## Functions

- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`

- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_add_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub (volatile atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`

- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub (atomic< _ITp * > *__a, ptrdiff_t __d) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub_explicit (volatile atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp * std::atomic_fetch_sub_explicit (atomic< _ITp * > *__a, ptrdiff_t __d, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_init (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_init (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store (atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store (volatile atomic< _ITp > *__a, _ITp __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept`

## 6.21.1 Detailed Description

This is a Standard C++ Library header.

## 6.22 atomic\_base.h File Reference

## Classes

- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)
- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)
- struct [std::\\_\\_atomic\\_base< \\_PTp \\* >](#)
- struct [std::\\_\\_atomic\\_flag\\_base](#)
- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic\\_flag](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_ALWAYS_INLINE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_VAR_INIT(_VI)`

## Typedefs

- typedef unsigned char [std::\\_\\_atomic\\_flag\\_data\\_type](#)
- typedef enum [std::memory\\_order](#) [std::memory\\_order](#)

## Enumerations

- enum [\\_\\_memory\\_order\\_modifier](#) { [\\_\\_memory\\_order\\_mask](#), [\\_\\_memory\\_order\\_modifier\\_mask](#), [\\_\\_memory\\_order\\_hle\\_acquire](#), [\\_\\_memory\\_order\\_hle\\_release](#) }
- enum [std::memory\\_order](#) { [memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#), [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

## Functions

- `std::__attribute ((__always_inline__)) void atomic_thread_fence(memory_order __m) noexcept`
- `constexpr memory_order std::__cmpexch_failure_order (memory_order __m) noexcept`
- `constexpr memory_order std::__cmpexch_failure_order2 (memory_order __m) noexcept`
- `template<typename _Tp >  
_Tp std::kill_dependency (_Tp __y) noexcept`
- `constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod)`
- `constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod)`

### 6.22.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 6.23 `atomic_futex.h` File Reference

### Namespaces

- [std](#)

### 6.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.24 `atomic_lockfree_defines.h` File Reference

### Macros

- #define [ATOMIC\\_BOOL\\_LOCK\\_FREE](#)
- #define **ATOMIC\_CHAR16\_T\_LOCK\_FREE**
- #define **ATOMIC\_CHAR32\_T\_LOCK\_FREE**
- #define **ATOMIC\_CHAR\_LOCK\_FREE**
- #define **ATOMIC\_INT\_LOCK\_FREE**
- #define **ATOMIC\_LLONG\_LOCK\_FREE**
- #define **ATOMIC\_LONG\_LOCK\_FREE**
- #define **ATOMIC\_POINTER\_LOCK\_FREE**
- #define **ATOMIC\_SHORT\_LOCK\_FREE**
- #define **ATOMIC\_WCHAR\_T\_LOCK\_FREE**

### 6.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 6.25 `atomic_word.h` File Reference

### Macros

- #define **\_GLIBCXX\_READ\_MEM\_BARRIER**
- #define **\_GLIBCXX\_WRITE\_MEM\_BARRIER**

## Typedefs

- typedef int **\_\_Atomic\_word**

## 6.25.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.26 atomicity.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- #define **\_GLIBCXX\_READ\_MEM\_BARRIER**
- #define **\_GLIBCXX\_WRITE\_MEM\_BARRIER**

## Functions

- static void **\_\_gnu\_cxx::\_\_atomic\_add\_single** (**\_Atomic\_word** \*\_\_mem, int \_\_val)
- else **\_\_gnu\_cxx::\_\_atomic\_add\_single** (\_\_mem, \_\_val)
- **\_Atomic\_word** **\_\_gnu\_cxx::\_\_attribute\_\_** ((\_\_unused\_\_)) **\_\_exchange\_and\_add**(volatile **\_Atomic\_word** \*
- static **\_Atomic\_word** **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (**\_Atomic\_word** \*\_\_mem, int \_\_val)
- else return **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (\_\_mem, \_\_val)
- **\_Atomic\_word** int **\_\_gnu\_cxx::throw** ()

## Variables

- static **\_Atomic\_word** int **\_\_gnu\_cxx::\_\_val**

## 6.26.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.27 auto\_ptr.h File Reference

## Classes

- class [std::auto\\_ptr<\\_Tp>](#)
- struct [std::auto\\_ptr\\_ref<\\_Tp1>](#)



## Namespaces

- [std](#)

### 6.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.28 `backward_warning.h` File Reference

### 6.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.29 `balanced_quicksort.h` File Reference

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter >](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer](#) (\_QSBThreadLocal<\_RAIter > \*\_\_tls, \_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_iam, \_ThreadIndex \_\_num\_threads, bool \_\_parent\_wait)
- template<typename \_RAIter, typename \_Compare >  
std::iterator\_traits<\_RAIter >::difference\_type [\\_\\_gnu\\_parallel::\\_\\_qsb\\_divide](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping](#) (\_QSBThreadLocal<\_RAIter > \*\_\_tls, \_Compare &\_\_comp, \_ThreadIndex \_\_iam, bool \_\_wait)

### 6.29.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.

It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

## 6.30 **base.h** File Reference

### Namespaces

- [\\_\\_gnu\\_profile](#)
- [std](#)
- [std::\\_\\_profile](#)

### 6.30.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

## 6.31 **base.h** File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st< \\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd< \\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate< \\_Predicate, argument\\_type >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess< \\_T1, \\_T2, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo< \\_T1, \\_T2 >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Less< \\_T1, \\_T2 >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Multiplies< \\_Tp1, \\_Tp2, \\_Result >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Plus< \\_Tp1, \\_Tp2, \\_Result >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence< \\_Tp, \\_DifferenceTp >](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequenceIterator< \\_Tp, \\_DifferenceTp >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)
- [\\_\\_gnu\\_sequential](#)
- [std](#)
- [std::\\_\\_parallel](#)

## Macros

- `#define _GLIBCXX_PARALLEL_ASSERT(_Condition)`

## Functions

- `void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b)`
- `_CASable __gnu_parallel::__encode2 (int __a, int __b)`
- `_ThreadIndex __gnu_parallel::__get_max_threads ()`
- `bool __gnu_parallel::__is_parallel (const _Parallelism __p)`
- `template<typename _RAIter, typename _Compare >  
_RAIter __gnu_parallel::__median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- `template<typename _Size >  
_Size __gnu_parallel::__rd_log2 (_Size __n)`
- `template<typename _Tp >  
const _Tp & __gnu_parallel::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >  
const _Tp & __gnu_parallel::min (const _Tp &__a, const _Tp &__b)`

### 6.31.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

## 6.32 `basic_file.h` File Reference

### Namespaces

- `std`

### 6.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.33 `basic_ios.h` File Reference

### Classes

- class `std::basic_ios<_CharT, _Traits >`

### Namespaces

- `std`

## Functions

- `template<typename _Facet >`  
`const _Facet & std::__check_facet (const _Facet *__f)`

### 6.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.34 `basic_ios.tcc` File Reference

## Namespaces

- `std`

## Macros

- `#define _BASIC_IOS_TCC`

### 6.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.35 `basic_iterator.h` File Reference

### 6.35.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 6.36 `basic_string.h` File Reference

## Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`
- struct `std::hash< string >`
- struct `std::hash< u16string >`
- struct `std::hash< u32string >`
- struct `std::hash< wstring >`

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_string_udls`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _↵  
_CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _↵  
_CharT, _Traits, _Alloc > &__str)`
- `template<>  
basic_istream< char > & std::getline (basic_istream< char > &__in, basic_string< char > &__str, char __↵  
delim)`
- `template<>  
basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str,  
wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,  
_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< char > std::literals::string\_literals::operator""s (const char  
*__str, size_t __len)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< wchar_t > std::literals::string\_literals::operator""s (const  
wchar_t *__str, size_t __len)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< char16_t > std::literals::string\_literals::operator""s (const  
char16_t *__str, size_t __len)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< char32_t > std::literals::string\_literals::operator""s (const  
char32_t *__str, size_t __len)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,  
const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _↵  
_Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ ( _CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ ( _CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT >`  
`_gnu_cxx::enable_if< __is_char< _CharT >::value, bool >::type std::operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs) noexcept`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)`  
`noexcept(/*conditional */)`

### 6.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.37 `basic_string.tcc` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _BASIC_STRING_TCC`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`

## 6.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.38 bin\_search\_tree\_.hpp File Reference

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

## 6.38.1 Detailed Description

Contains an implementation class for binary search tree.

## 6.39 binary\_heap\_.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >



## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

### 6.39.1 Detailed Description

Contains an implementation class for a binary heap.

## 6.40 binders.h File Reference

### Classes

- class [std::binder1st<\\_Operation>](#)
- class [std::binder2nd<\\_Operation>](#)

### Namespaces

- [std](#)

### Functions

- `template<typename _Operation, typename _Tp>`  
`binder1st<_Operation> std::bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp>`  
`binder2nd<_Operation> std::bind2nd (const _Operation &__fn, const _Tp &__x)`

### 6.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.41 binomial\_heap.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap<Value\\_Type, Cmp\\_Fn, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 6.41.1 Detailed Description

Contains an implementation class for a binomial heap.

## 6.42 binomial\_heap\_base.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_ASSERT_VALID_COND(X, _StrictlyBinomial)`
- `#define PB_DS_B_HEAP_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 6.42.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 6.43 bitmap\_allocator.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector< \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Bitmap\\_counter< \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Ffit\\_finder< \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::free\\_list](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::\\_\\_detail](#)

## Macros

- `#define \_BALLOC\_ALIGN\_BYTES`

## Enumerations

- `enum { bits\_per\_byte, bits\_per\_block }`

## Functions

- `void \_\_gnu\_cxx::\_\_detail::\_\_bit\_allocate (size_t *__pmap, size_t __pos) throw ()`
- `void \_\_gnu\_cxx::\_\_detail::\_\_bit\_free (size_t *__pmap, size_t __pos) throw ()`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >  
_ForwardIterator \_\_gnu\_cxx::\_\_detail::\_\_lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const  
_Tp &__val, _Compare __comp)`
- `template<typename _AddrPair >  
size_t \_\_gnu\_cxx::\_\_detail::\_\_num\_bitmaps (_AddrPair __ap)`
- `template<typename _AddrPair >  
size_t \_\_gnu\_cxx::\_\_detail::\_\_num\_blocks (_AddrPair __ap)`
- `size_t \_\_gnu\_cxx::\_\_Bit\_scan\_forward (size_t __num)`
- `template<typename _Tp1, typename _Tp2 >  
bool \_\_gnu\_cxx::operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _Tp1, typename _Tp2 >  
bool \_\_gnu\_cxx::operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`

### 6.43.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 6.43.2 Macro Definition Documentation

#### 6.43.2.1 [\\_BALLOC\\_ALIGN\\_BYTES](#)

```
#define \_BALLOC\_ALIGN\_BYTES
```

The constant in the expression below is the alignment required in bytes.

Definition at line 43 of file `bitmap_allocator.h`.

6.44 `bitset` File Reference

## Classes

- struct `std::_Base_bitset<_Nw>`
- struct `std::_Base_bitset<0>`
- struct `std::_Base_bitset<1>`
- class `std::bitset<_Nb>`
- class `std::bitset<_Nb>::reference`
- struct `std::hash<::bitset<_Nb>>`

## Namespaces

- `std`

## Macros

- `#define _GLIBCXX_BITSET`
- `#define _GLIBCXX_BITSET_BITS_PER_ULL`
- `#define _GLIBCXX_BITSET_BITS_PER_WORD`
- `#define _GLIBCXX_BITSET_WORDS(__n)`

## Functions

- `template<size_t _Nb>`  
`bitset<_Nb> std::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<size_t _Nb>`  
`bitset<_Nb> std::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<size_t _Nb>`  
`bitset<_Nb> std::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`

## 6.44.1 Detailed Description

This is a Standard C++ Library header.

## 6.45 bitset File Reference

### Classes

- class [std::\\_\\_debug::bitset<\\_Nb>](#)
- struct [std::hash<\\_\\_debug::bitset<\\_Nb>>](#)

### Namespaces

- [std](#)
- [std::\\_\\_debug](#)

### Functions

- `template<size_t _Nb>  
bitset<_Nb> std::__debug::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_ostream<_CharT, _Traits> & std::__debug::operator<< (std::basic\_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_istream<_CharT, _Traits> & std::__debug::operator>> (std::basic\_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>  
bitset<_Nb> std::__debug::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<size_t _Nb>  
bitset<_Nb> std::__debug::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`

#### 6.45.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.46 bitset File Reference

### Classes

- class [std::\\_\\_profile::bitset<\\_Nb>](#)
- struct [std::hash<\\_\\_profile::bitset<\\_Nb>>](#)

### Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & std::__profile::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & std::__profile::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`

## 6.46.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.47 bool\_set File Reference

## Classes

- class `std::tr2::bool_set`

## Namespaces

- `std`
- `std::tr2`

## Macros

- `#define _GLIBCXX_TR2_BOOL_SET`

## Functions

- `bool std::tr2::certainly (bool_set __b)`
- `bool std::tr2::contains (bool_set __s, bool_set __t)`
- `bool std::tr2::equals (bool_set __s, bool_set __t)`
- `bool std::tr2::is_emptyset (bool_set __b)`
- `bool std::tr2::is_indeterminate (bool_set __b)`
- `bool std::tr2::is_singleton (bool_set __b)`
- `bool_set std::tr2::operator!= (bool __s, bool_set __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool __s, bool_set __t)`

- `bool_set std::tr2::operator& (bool_set __s, bool __t)`
- `bool_set std::tr2::operator== (bool __s, bool_set __t)`
- `bool_set std::tr2::operator== (bool_set __s, bool __t)`
- `bool_set std::tr2::operator^ (bool __s, bool_set __t)`
- `bool_set std::tr2::operator^ (bool_set __s, bool __t)`
- `bool_set std::tr2::operator| (bool __s, bool_set __t)`
- `bool_set std::tr2::operator| (bool_set __s, bool __t)`
- `bool std::tr2::possibly (bool_set __b)`
- `bool_set std::tr2::set_complement (bool_set __b)`
- `bool_set std::tr2::set_intersection (bool __s, bool_set __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool_set __t)`
- `bool_set std::tr2::set_union (bool __s, bool_set __t)`
- `bool_set std::tr2::set_union (bool_set __s, bool __t)`
- `bool_set std::tr2::set_union (bool_set __s, bool_set __t)`

#### 6.47.1 Detailed Description

This is a TR2 C++ Library header.

## 6.48 `bool_set.tcc` File Reference

### Namespaces

- [std](#)
- [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_BOOL_SET_TCC`

#### 6.48.1 Detailed Description

This is a TR2 C++ Library header.

## 6.49 `boost_concept_check.h` File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

## Functions

- `template<class _Tp >`  
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`  
`void __gnu_cxx::__function_requires ()`

### 6.49.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.50 branch\_policy.hpp File Reference

### Classes

- [struct \*\*\\_\\_gnu\\_pbds::detail::branch\\_policy\*\*< Node\\_Cltr, Node\\_Itr, \\_Alloc >](#)
- [struct \*\*\\_\\_gnu\\_pbds::detail::branch\\_policy\*\*< Node\\_Cltr, Node\\_Cltr, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.50.1 Detailed Description

Contains a base class for branch policies.

## 6.51 c++0x\_warning.h File Reference

### 6.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.



## 6.52 c++allocator.h File Reference

### Namespaces

- [std](#)

### Typedefs

- `template<typename _Tp >`  
`using std::\_\_allocator\_base = \_\_gnu\_cxx::new\_allocator<_Tp >`

### 6.52.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.53 c++config.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define` `__GLIBCXX__`
- `#define` `__glibcxx_assert`(\_Condition)
- `#define` `__N`(msgid)
- `#define` `_GLIBCXX11_USE_C99_MATH`
- `#define` `_GLIBCXX11_USE_C99_STDIO`
- `#define` `_GLIBCXX11_USE_C99_STDLIB`
- `#define` `_GLIBCXX11_USE_C99_WCHAR`
- `#define` `_GLIBCXX14_CONSTEXPR`
- `#define` `_GLIBCXX17_CONSTEXPR`
- `#define` `_GLIBCXX17_DEPRECATED`
- `#define` `_GLIBCXX17_INLINE`
- `#define` `_GLIBCXX98_USE_C99_COMPLEX`
- `#define` `_GLIBCXX98_USE_C99_MATH`
- `#define` `_GLIBCXX98_USE_C99_STDIO`
- `#define` `_GLIBCXX98_USE_C99_STDLIB`
- `#define` `_GLIBCXX98_USE_C99_WCHAR`
- `#define` `_GLIBCXX_ABI_TAG_CXX11`
- `#define` `_GLIBCXX_ATOMIC_BUILTINS`
- `#define` `_GLIBCXX_BEGIN_EXTERN_C`
- `#define` `_GLIBCXX_BEGIN_NAMESPACE_ALGO`
- `#define` `_GLIBCXX_BEGIN_NAMESPACE_CONTAINER`

- `#define _GLIBCXX_BEGIN_NAMESPACE_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_DEFAULT_ABI_TAG`
- `#define _GLIBCXX_DEPRECATED`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_NAMESPACE_ALGO`
- `#define _GLIBCXX_END_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_END_NAMESPACE_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_LDBL`
- `#define _GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_VERSION`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_FAST_MATH`
- `#define _GLIBCXX_FULLY_DYNAMIC_STRING`
- `#define _GLIBCXX_HAVE__CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_HAVE_ACOSF`
- `#define _GLIBCXX_HAVE_ACOSL`
- `#define _GLIBCXX_HAVE_ALIGNED_ALLOC`
- `#define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- `#define _GLIBCXX_HAVE_ASINF`
- `#define _GLIBCXX_HAVE_ASINL`
- `#define _GLIBCXX_HAVE_AT_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_ATAN2F`
- `#define _GLIBCXX_HAVE_ATAN2L`
- `#define _GLIBCXX_HAVE_ATANF`
- `#define _GLIBCXX_HAVE_ATANL`
- `#define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- `#define _GLIBCXX_HAVE_CEILF`
- `#define _GLIBCXX_HAVE_CEILL`
- `#define _GLIBCXX_HAVE_COMPLEX_H`
- `#define _GLIBCXX_HAVE_COSF`
- `#define _GLIBCXX_HAVE_COSHF`
- `#define _GLIBCXX_HAVE_COSHL`
- `#define _GLIBCXX_HAVE_COSL`
- `#define _GLIBCXX_HAVE_DIRENT_H`
- `#define _GLIBCXX_HAVE_DLFCN_H`
- `#define _GLIBCXX_HAVE_EBADMSG`
- `#define _GLIBCXX_HAVE_ECANCELED`
- `#define _GLIBCXX_HAVE_ECHILD`
- `#define _GLIBCXX_HAVE_EIDRM`
- `#define _GLIBCXX_HAVE_ENDIAN_H`
- `#define _GLIBCXX_HAVE_ENODATA`
- `#define _GLIBCXX_HAVE_ENOLINK`
- `#define _GLIBCXX_HAVE_ENOSPC`
- `#define _GLIBCXX_HAVE_ENOSR`
- `#define _GLIBCXX_HAVE_ENOSTR`
- `#define _GLIBCXX_HAVE_ENOTRECOVERABLE`
- `#define _GLIBCXX_HAVE_ENOTSUP`
- `#define _GLIBCXX_HAVE_EOVERFLOW`

- `#define _GLIBCXX_HAVE_EOWNERDEAD`
- `#define _GLIBCXX_HAVE_EPERM`
- `#define _GLIBCXX_HAVE_EPROTO`
- `#define _GLIBCXX_HAVE_ETIME`
- `#define _GLIBCXX_HAVE_ETIMEDOUT`
- `#define _GLIBCXX_HAVE_ETXTBSY`
- `#define _GLIBCXX_HAVE_EWOULDBLOCK`
- `#define _GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46`
- `#define _GLIBCXX_HAVE_EXECINFO_H`
- `#define _GLIBCXX_HAVE_EXPF`
- `#define _GLIBCXX_HAVE_EXPL`
- `#define _GLIBCXX_HAVE_FABSF`
- `#define _GLIBCXX_HAVE_FABSL`
- `#define _GLIBCXX_HAVE_FCNTL_H`
- `#define _GLIBCXX_HAVE_FENV_H`
- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOAT_H`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_GETIPINFO`
- `#define _GLIBCXX_HAVE_GETS`
- `#define _GLIBCXX_HAVE_HYPOT`
- `#define _GLIBCXX_HAVE_HYPOTF`
- `#define _GLIBCXX_HAVE_HYPOTL`
- `#define _GLIBCXX_HAVE_ICONv`
- `#define _GLIBCXX_HAVE_INT64_T`
- `#define _GLIBCXX_HAVE_INT64_T_LONG`
- `#define _GLIBCXX_HAVE_INTTYPES_H`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`
- `#define _GLIBCXX_HAVE_LINUX_RANDOM_H`

- `#define _GLIBCXX_HAVE_LINUX_TYPES_H`
- `#define _GLIBCXX_HAVE_LOCALE_H`
- `#define _GLIBCXX_HAVE_LOG10F`
- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_MBSTATE_T`
- `#define _GLIBCXX_HAVE_MEMALIGN`
- `#define _GLIBCXX_HAVE_MEMORY_H`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POSIX_MEMALIGN`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`
- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`
- `#define _GLIBCXX_HAVE_SINHL`
- `#define _GLIBCXX_HAVE_SINL`
- `#define _GLIBCXX_HAVE_SQRTF`
- `#define _GLIBCXX_HAVE_SQRTL`
- `#define _GLIBCXX_HAVE_STDALIGN_H`
- `#define _GLIBCXX_HAVE_STDBOOL_H`
- `#define _GLIBCXX_HAVE_STDINT_H`
- `#define _GLIBCXX_HAVE_STDLIB_H`
- `#define _GLIBCXX_HAVE_STRERROR_L`
- `#define _GLIBCXX_HAVE_STRERROR_R`
- `#define _GLIBCXX_HAVE_STRING_H`
- `#define _GLIBCXX_HAVE_STRINGS_H`
- `#define _GLIBCXX_HAVE_STRTOF`
- `#define _GLIBCXX_HAVE_STRTOLD`
- `#define _GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE`
- `#define _GLIBCXX_HAVE_STRXFRM_L`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_SYS_IOCTL_H`
- `#define _GLIBCXX_HAVE_SYS_IPC_H`
- `#define _GLIBCXX_HAVE_SYS_PARAM_H`
- `#define _GLIBCXX_HAVE_SYS_RESOURCE_H`
- `#define _GLIBCXX_HAVE_SYS_SEM_H`
- `#define _GLIBCXX_HAVE_SYS_STAT_H`
- `#define _GLIBCXX_HAVE_SYS_STATVFS_H`
- `#define _GLIBCXX_HAVE_SYS_SYSINFO_H`
- `#define _GLIBCXX_HAVE_SYS_TIME_H`
- `#define _GLIBCXX_HAVE_SYS_TYPES_H`
- `#define _GLIBCXX_HAVE_SYS_UIO_H`

- `#define _GLIBCXX_HAVE_TANF`
- `#define _GLIBCXX_HAVE_TANHF`
- `#define _GLIBCXX_HAVE_TANHL`
- `#define _GLIBCXX_HAVE_TANL`
- `#define _GLIBCXX_HAVE_TGMATH_H`
- `#define _GLIBCXX_HAVE_TLS`
- `#define _GLIBCXX_HAVE_UCHAR_H`
- `#define _GLIBCXX_HAVE_UNISTD_H`
- `#define _GLIBCXX_HAVE_UTIME_H`
- `#define _GLIBCXX_HAVE_VFWSCANF`
- `#define _GLIBCXX_HAVE_VSWSCANF`
- `#define _GLIBCXX_HAVE_VWSCANF`
- `#define _GLIBCXX_HAVE_WCHAR_H`
- `#define _GLIBCXX_HAVE_WCSTOF`
- `#define _GLIBCXX_HAVE_WCTYPE_H`
- `#define _GLIBCXX_HAVE_WRITEV`
- `#define _GLIBCXX_HOSTED`
- `#define _GLIBCXX_ICONV_CONST`
- `#define _GLIBCXX_INLINE_VERSION`
- `#define _GLIBCXX_MANGLE_SIZE_T`
- `#define _GLIBCXX_NAMESPACE_CXX11`
- `#define _GLIBCXX_NAMESPACE_LDBL`
- `#define _GLIBCXX_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_NOEXCEPT_PARM`
- `#define _GLIBCXX_NOEXCEPT_QUAL`
- `#define _GLIBCXX_PACKAGE__GLIBCXX_VERSION`
- `#define _GLIBCXX_PACKAGE_BUGREPORT`
- `#define _GLIBCXX_PACKAGE_NAME`
- `#define _GLIBCXX_PACKAGE_STRING`
- `#define _GLIBCXX_PACKAGE_TARNAME`
- `#define _GLIBCXX_PACKAGE_URL`
- `#define _GLIBCXX_PSEUDO_VISIBILITY(V)`
- `#define _GLIBCXX_RELEASE`
- `#define _GLIBCXX_RES_LIMITS`
- `#define _GLIBCXX_STD_A`
- `#define _GLIBCXX_STD_C`
- `#define _GLIBCXX_STDIO_EOF`
- `#define _GLIBCXX_STDIO_SEEK_CUR`
- `#define _GLIBCXX_STDIO_SEEK_END`
- `#define _GLIBCXX_SYMVER`
- `#define _GLIBCXX_SYMVER_GNU`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER(A)`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE(A)`
- `#define _GLIBCXX_THROW_OR_ABORT(_EXC)`
- `#define _GLIBCXX_TXN_SAFE`
- `#define _GLIBCXX_TXN_SAFE_DYN`
- `#define _GLIBCXX_USE_ALLOCATOR_NEW`
- `#define _GLIBCXX_USE_C11_UCHAR_CXX11`
- `#define _GLIBCXX_USE_C99`
- `#define _GLIBCXX_USE_C99_COMPLEX`
- `#define _GLIBCXX_USE_C99_COMPLEX_TR1`

- #define `_GLIBCXX_USE_C99_CTYPE_TR1`
- #define `_GLIBCXX_USE_C99_FENV_TR1`
- #define `_GLIBCXX_USE_C99_INTTYPES_TR1`
- #define `_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- #define `_GLIBCXX_USE_C99_MATH`
- #define `_GLIBCXX_USE_C99_MATH_TR1`
- #define `_GLIBCXX_USE_C99_STDINT_TR1`
- #define `_GLIBCXX_USE_C99_STDIO`
- #define `_GLIBCXX_USE_C99_STDLIB`
- #define `_GLIBCXX_USE_C99_WCHAR`
- #define `_GLIBCXX_USE_CLOCK_MONOTONIC`
- #define `_GLIBCXX_USE_CLOCK_REALTIME`
- #define `_GLIBCXX_USE_CXX11_ABI`
- #define `_GLIBCXX_USE_DECIMAL_FLOAT`
- #define `_GLIBCXX_USE_DEPRECATED`
- #define `_GLIBCXX_USE_DUAL_ABI`
- #define `_GLIBCXX_USE_FCHMOD`
- #define `_GLIBCXX_USE_FCHMODAT`
- #define `_GLIBCXX_USE_GET_NPROCS`
- #define `_GLIBCXX_USE_GETTIMEOFDAY`
- #define `_GLIBCXX_USE_INT128`
- #define `_GLIBCXX_USE_LFS`
- #define `_GLIBCXX_USE_LONG_LONG`
- #define `_GLIBCXX_USE_NANOSLEEP`
- #define `_GLIBCXX_USE_RANDOM_TR1`
- #define `_GLIBCXX_USE_REALPATH`
- #define `_GLIBCXX_USE_SC_NPROCESSORS_ONLN`
- #define `_GLIBCXX_USE_SCHED_YIELD`
- #define `_GLIBCXX_USE_SENDFILE`
- #define `_GLIBCXX_USE_ST_MTIM`
- #define `_GLIBCXX_USE_TMPNAM`
- #define `_GLIBCXX_USE_UTIMENSAT`
- #define `_GLIBCXX_USE_WCHAR_T`
- #define `_GLIBCXX_USE_WEAK_REF`
- #define `_GLIBCXX_VERBOSE`
- #define `_GLIBCXX_VISIBILITY(V)`
- #define `_GLIBCXX_WEAK_DEFINITION`
- #define `_GLIBCXX_X86_RDRAND`
- #define `_GTHREAD_USE_MUTEX_TIMEDLOCK`
- #define `LT_OBJDIR`
- #define `STDC_HEADERS`

#### Typedefs

- typedef `__PTRDIFF_TYPE__` `std::ptrdiff_t`
- typedef `__SIZE_TYPE__` `std::size_t`

#### Functions

- namespace `__cxx11` `std::` `__attribute__((abi_tag("cxx11")))`
- namespace `__cxx11` `gnu_cxx::` `__attribute__((abi_tag("cxx11")))`

#### Variables

- `decltype(nullptr)` typedef **`std::nullptr_t`**

##### 6.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.54 `c++io.h` File Reference

#### Namespaces

- [`std`](#)

#### Typedefs

- typedef FILE **`std::__c_file`**
- typedef `__pthread_mutex_t` **`std::__c_lock`**

##### 6.54.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.55 `c++locale.h` File Reference

#### Namespaces

- [`std`](#)

#### Macros

- `#define` **`_GLIBCXX_C_LOCALE_GNU`**
- `#define` **`_GLIBCXX_NUM_CATEGORIES`**

#### Typedefs

- typedef `__locale_t` **`std::__c_locale`**

## Functions

- `int std::__convert_from_v` (`const __c_locale &__cloc __attribute__((__unused__))`, `char *__out`, `const int __↵`  
`size __attribute__((__unused__))`, `const char *__fmt,...`)

## 6.55.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

6.56 `c++locale_internal.h` File Reference

## Namespaces

- [std](#)

## Functions

- Catalogs & `std::get_catalogs ()`

## 6.56.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

6.57 `cassert` File Reference

## 6.57.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.58 `cast.h` File Reference

## Classes

- `struct __gnu_cxx::Caster<_ToType >`



## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (_FromType *__arg)`

### 6.58.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

## 6.59 cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp File Reference

### 6.59.1 Detailed Description

Contains a resize trigger implementation.

## 6.60 cc\_ht\_map.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CC_HASH_NAME`
- `#define PB_DS_CC_HASH_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_HASH_FN_C_DEC`

## 6.60.1 Detailed Description

Contains an implementation class for `cc_ht_map_`.

## 6.61 ccomplex File Reference

## Macros

- `#define _GLIBCXX_CCOMPLEX`

## 6.61.1 Detailed Description

This is a Standard C++ Library header.

## 6.62 ccomplex File Reference

## Macros

- `#define _GLIBCXX_TR1_CCOMPLEX`

## 6.62.1 Detailed Description

This is a TR1 C++ Library header.

## 6.63 ctype File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_CCTYPE`

### 6.63.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `cctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.64 cctype File Reference

## Macros

- `#define _GLIBCXX_TR1_CCTYPE`

### 6.64.1 Detailed Description

This is a TR1 C++ Library header.

## 6.65 cerrno File Reference

## Macros

- `#define _GLIBCXX_CERRNO`
- `#define errno`

### 6.65.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.66 cenv File Reference

## Macros

- `#define _GLIBCXX_CFENV`

### 6.66.1 Detailed Description

This is a Standard C++ Library header.

## 6.67 **cfenv File Reference**

### Macros

- `#define _GLIBCXX_TR1_CFENV`

### 6.67.1 Detailed Description

This is a TR1 C++ Library header.

## 6.68 **cfloat File Reference**

### Macros

- `#define _GLIBCXX_CFLOAT`
- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 6.68.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.69 **cfloat File Reference**

### Macros

- `#define _GLIBCXX_TR1_CFLOAT`

### 6.69.1 Detailed Description

This is a TR1 C++ Library header.

## 6.70 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_Char\\_types<\\_CharT>](#)
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)
- struct [std::char\\_traits<\\_CharT>](#)
- struct [std::char\\_traits<char>](#)
- struct [std::char\\_traits<wchar\\_t>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define _GLIBCXX_ALWAYS_INLINE`

#### 6.70.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 6.71 checkers.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename [\\_Iter](#) , typename [\\_Compare](#) >  
bool [\\_\\_gnu\\_parallel::\\_\\_is\\_sorted](#) ([\\_Iter](#) \_\_begin, [\\_Iter](#) \_\_end, [\\_Compare](#) \_\_comp)

#### 6.71.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

## 6.72 chrono File Reference

## Classes

- struct [std::chrono::\\_V2::steady\\_clock](#)
- struct [std::chrono::\\_V2::system\\_clock](#)
- struct [std::chrono::duration< \\_Rep, \\_Period >](#)
- struct [std::chrono::duration< \\_Rep, \\_Period >](#)
- struct [std::chrono::duration\\_values< \\_Rep >](#)
- struct [std::chrono::time\\_point< \\_Clock, \\_Dur >](#)
- struct [std::chrono::time\\_point< \\_Clock, \\_Dur >](#)
- struct [std::chrono::treat\\_as\\_floating\\_point< \\_Rep >](#)

## Namespaces

- [std](#)
- [std::chrono](#)

## Macros

- `#define __cpp_lib_chrono_udls`
- `#define _GLIBCXX_CHRONO`

## Typedefs

- `template<typename _Rep1, typename _Rep2, typename _CRep = typename common_type<_Rep1, _Rep2>::type>  
using std::chrono::\_\_common\_rep\_t = typename enable_if< is_convertible< const _Rep2 &, _CRep >::value,  
_CRep >::type`
- `template<typename _Tp >  
using std::chrono::\_\_disable\_if\_is\_duration = typename enable_if<!__is_duration< _Tp >::value, _Tp >↵  
::type`
- `template<typename _Tp >  
using std::chrono::\_\_enable\_if\_is\_duration = typename enable_if< __is_duration< _Tp >::value, _Tp >::type`
- `using std::chrono::\_V2::high\_resolution\_clock = system_clock`
- `typedef duration< int64_t, ratio< 3600 > > std::chrono::hours`
- `typedef duration< int64_t, micro > std::chrono::microseconds`
- `typedef duration< int64_t, milli > std::chrono::milliseconds`
- `typedef duration< int64_t, ratio< 60 > > std::chrono::minutes`
- `typedef duration< int64_t, nano > std::chrono::nanoseconds`
- `typedef duration< int64_t > std::chrono::seconds`

## Functions

- `template<typename _Dur, char... _Digits>`  
`constexpr _Dur std::literals::chrono_literals::__check_overflow ()`
- `template<typename _ToDur, typename _Rep, typename _Period >`  
`constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &←  
__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator!= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,  
_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator!= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,  
_Dur2 > &__rhs)`
- `constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long  
double __hours)`
- `template<char... _Digits>`  
`constexpr chrono::hours std::literals::chrono_literals::operator""h ()`
- `constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long  
double __mins)`
- `template<char... _Digits>`  
`constexpr chrono::minutes std::literals::chrono_literals::operator""min ()`
- `constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double ←  
__msecs)`
- `template<char... _Digits>`  
`constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()`
- `constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double  
__nsecs)`
- `template<char... _Digits>`  
`constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()`
- `constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)`
- `template<char... _Digits>`  
`constexpr chrono::seconds std::literals::chrono_literals::operator""s ()`
- `constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double  
__usecs)`
- `template<char... _Digits>`  
`constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::←  
::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono←  
::operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > std::chrono::operator* (const duration<  
_Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Rep2, typename _Period >`  
`constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1  
&__s, const duration< _Rep2, _Period > &__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono←  
::operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 >::type > std::`  
`::chrono::operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2::type > std::`  
`::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::`  
`::operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 >::type > std::`  
`::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 >, _Period > std::`  
`::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator== (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator>= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _`  
`Clock, _Dur2 > &__rhs)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`  
`constexpr enable_if< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > >::type std::chrono::time_point_cast`  
`(const time_point< _Clock, _Dur > &__t)`



### 6.72.1 Detailed Description

This is a Standard C++ Library header.

### 6.72.2 Typedef Documentation

#### 6.72.2.1 high\_resolution\_clock

```
using std::chrono::\_V2::high\_resolution\_clock = typedef system_clock
```

Highest-resolution clock.

This is the clock "with the shortest tick period." Alias to `std::system_clock` until higher-than-nanosecond definitions become feasible.

Definition at line 878 of file `chrono`.

## 6.73 chrono File Reference

### Namespaces

- [std](#)
- [std::chrono](#)

### Macros

- `#define \_GLIBCXX\_EXPERIMENTAL\_CHRONO`

### Variables

- `template<typename _Rep >`  
`constexpr bool std::chrono::experimental::fundamentals\_v1::treat\_as\_floating\_point\_v`

### 6.73.1 Detailed Description

This is a TS C++ Library header.

## 6.74 cinttypes File Reference

### Macros

- `#define \_GLIBCXX\_CINTTYPES`

#### 6.74.1 Detailed Description

This is a Standard C++ Library header.

## 6.75 cinttypes File Reference

### Macros

- `#define _GLIBCXX_TR1_CINTTYPES`

#### 6.75.1 Detailed Description

This is a TR1 C++ Library header.

## 6.76 ciso646 File Reference

#### 6.76.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, which is empty in C++.

## 6.77 climits File Reference

### Macros

- `#define _GLIBCXX_CLIMITS`
- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

#### 6.77.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.78 climits File Reference

### Macros

- `#define _GLIBCXX_TR1_CLIMITS`

### 6.78.1 Detailed Description

This is a TR1 C++ Library header.

## 6.79 clocale File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CLOCALE`

### 6.79.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.80 cmath File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CMATH`
- `#define _GLIBCXX_INCLUDE_NEXT_C_HEADERS`

## Functions

- constexpr float **std::acos** (float \_\_x)
- constexpr long double **std::acos** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::acos** (\_Tp \_\_x)
- constexpr float **std::asin** (float \_\_x)
- constexpr long double **std::asin** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::asin** (\_Tp \_\_x)
- constexpr float **std::atan** (float \_\_x)
- constexpr long double **std::atan** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::atan** (\_Tp \_\_x)
- constexpr float **std::atan2** (float \_\_y, float \_\_x)
- constexpr long double **std::atan2** (long double \_\_y, long double \_\_x)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::atan2** (\_Tp \_\_y, \_Up \_\_x)
- constexpr float **std::ceil** (float \_\_x)
- constexpr long double **std::ceil** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::ceil** (\_Tp \_\_x)
- constexpr float **std::cos** (float \_\_x)
- constexpr long double **std::cos** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::cos** (\_Tp \_\_x)
- constexpr float **std::cosh** (float \_\_x)
- constexpr long double **std::cosh** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::cosh** (\_Tp \_\_x)
- constexpr float **std::exp** (float \_\_x)
- constexpr long double **std::exp** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::exp** (\_Tp \_\_x)
- constexpr float **std::fabs** (float \_\_x)
- constexpr long double **std::fabs** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::fabs** (\_Tp \_\_x)
- constexpr float **std::floor** (float \_\_x)
- constexpr long double **std::floor** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::floor** (\_Tp \_\_x)
- constexpr float **std::fmod** (float \_\_x, float \_\_y)
- constexpr long double **std::fmod** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmod** (\_Tp \_\_x, \_Up \_\_y)
- float **std::frexp** (float \_\_x, int \*\_\_exp)
- long double **std::frexp** (long double \_\_x, int \*\_\_exp)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::frexp** (\_Tp \_\_x, int \*\_\_exp)
- constexpr float **std::ldexp** (float \_\_x, int \_\_exp)

- constexpr long double **std::ldexp** (long double \_\_x, int \_\_exp)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::ldexp** (\_Tp \_\_x, int \_\_exp)
- constexpr float **std::log** (float \_\_x)
- constexpr long double **std::log** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log** (\_Tp \_\_x)
- constexpr float **std::log10** (float \_\_x)
- constexpr long double **std::log10** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log10** (\_Tp \_\_x)
- float **std::modf** (float \_\_x, float \*\_\_iptr)
- long double **std::modf** (long double \_\_x, long double \*\_\_iptr)
- constexpr float **std::pow** (float \_\_x, float \_\_y)
- constexpr long double **std::pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::pow** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::sin** (float \_\_x)
- constexpr long double **std::sin** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::sin** (\_Tp \_\_x)
- constexpr float **std::sinh** (float \_\_x)
- constexpr long double **std::sinh** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::sinh** (\_Tp \_\_x)
- constexpr float **std::sqrt** (float \_\_x)
- constexpr long double **std::sqrt** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::sqrt** (\_Tp \_\_x)
- constexpr float **std::tan** (float \_\_x)
- constexpr long double **std::tan** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::tan** (\_Tp \_\_x)
- constexpr float **std::tanh** (float \_\_x)
- constexpr long double **std::tanh** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::tanh** (\_Tp \_\_x)

### 6.80.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.81 cmath File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_CMATH`

#### 6.81.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.82 cmath File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CMATH`

### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc\_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::tr1::assoc\_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc\_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::tr1::assoc\_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)`
- `float std::tr1::betaf (float __x, float __y)`
- `long double std::tr1::betal (long double __x, long double __y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_1 (_Tp __k)`
- `float std::tr1::comp\_ellint\_1f (float __k)`
- `long double std::tr1::comp\_ellint\_1l (long double __k)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_2 (_Tp __k)`

- float **std::tr1::comp\_ellint\_2f** (float \_\_k)
- long double **std::tr1::comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type **std::tr1::comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **std::tr1::comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **std::tr1::comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type **std::tr1::conf\_hyperg** (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::conf\_hypergf** (float \_\_a, float \_\_c, float \_\_x)
- long double **std::tr1::conf\_hypergl** (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_i** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_j** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_k** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- float **std::tr1::fabs** (float \_\_x)
- long double **std::tr1::fabsl** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::fabs** (\_Tp \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)

- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c,`  
`__Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`
- `long double std::tr1::hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)`
- `float std::tr1::laguerref (unsigned int __n, float __x)`
- `long double std::tr1::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)`
- `float std::tr1::legendref (unsigned int __n, float __x)`
- `long double std::tr1::legendrel (unsigned int __n, long double __x)`
- `float std::tr1::pow (float __x, float __y)`
- `long double std::tr1::pow (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type std::tr1::pow ( _Tp __x, _Up __y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann\_zeta ( _Tp __x)`
- `float std::tr1::riemann\_zetaf (float __x)`
- `long double std::tr1::riemann\_zetal (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_bessel (unsigned int __n, _Tp __x)`
- `float std::tr1::sph\_besself (unsigned int __n, float __x)`
- `long double std::tr1::sph\_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float std::tr1::sph\_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double std::tr1::sph\_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_neumann (unsigned int __n, _Tp __x)`
- `float std::tr1::sph\_neumannf (unsigned int __n, float __x)`
- `long double std::tr1::sph\_neumannl (unsigned int __n, long double __x)`

### 6.82.1 Detailed Description

This is a TR1 C++ Library header.

## 6.83 cmp\_fn\_imps.hpp File Reference

### 6.83.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container comparison related functions.

## 6.84 codecvt File Reference

### Namespaces

- [std](#)



## Macros

- `#define _GLIBCXX_CODECVT`
- `#define _GLIBCXX_CODECVT_SPECIALIZATION(_NAME, _ELEM)`
- `#define _GLIBCXX_CODECVT_SPECIALIZATION2(_NAME, _ELEM)`

## Enumerations

- enum `codecvt_mode` { `consume_header`, `generate_header`, `little_endian` }

### 6.84.1 Detailed Description

This is a Standard C++ Library header.

## 6.85 `codecvt.h` File Reference

### Classes

- class `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`
- class `std::codecvt< _InternT, _ExternT, _StateT >`
- class `std::codecvt< char, char, mbstate_t >`
- class `std::codecvt< char16_t, char, mbstate_t >`
- class `std::codecvt< char32_t, char, mbstate_t >`
- class `std::codecvt< wchar_t, char, mbstate_t >`
- class `std::codecvt_base`
- class `std::codecvt_byname< _InternT, _ExternT, _StateT >`

### Namespaces

- `std`

### 6.85.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.86 `codecvt_specializations.h` File Reference

### Classes

- struct `__gnu_cxx::encoding_char_traits< _CharT >`
- class `__gnu_cxx::encoding_state`
- class `std::codecvt< _InternT, _ExternT, encoding_state >`

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Functions

- `template<typename _Tp >  
size_t std::__iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **↵  
__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`

## 6.86.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.87 compatibility.h File Reference

## 6.87.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

## 6.88 compatibility.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Tp >  
_Tp __gnu_parallel::__add_omp (volatile _Tp *__ptr, _Tp __addend)`
- `template<typename _Tp >  
bool __gnu_parallel::__cas_omp (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp >  
bool \_\_gnu\_parallel::\_\_compare\_and\_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp >  
_Tp \_\_gnu\_parallel::\_\_fetch\_and\_add (volatile _Tp *__ptr, _Tp __addend)`
- `void \_\_gnu\_parallel::\_\_yield ()`

## 6.88.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.

This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

## 6.89 compiletime\_settings.h File Reference

### Macros

- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_PARALLEL_ASSERTIONS`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

### 6.89.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

### 6.89.2 Macro Definition Documentation

#### 6.89.2.1 \_GLIBCXX\_CALL

```
#define _GLIBCXX_CALL(  
    __n )
```

Macro to produce log message when entering a function.

#### Parameters

<code>_↔</code>	Input size.
<code>_n</code>	

#### See also

`_GLIBCXX_VERBOSE_LEVEL`

Definition at line 44 of file `compiletime_settings.h`.

#### 6.89.2.2 \_GLIBCXX\_PARALLEL\_ASSERTIONS

```
#define _GLIBCXX_PARALLEL_ASSERTIONS
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

### 6.89.2.3 `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the L1 cache for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `comPILEtime_settings.h`.

### 6.89.2.4 `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 74 of file `comPILEtime_settings.h`.

### 6.89.2.5 `_GLIBCXX_SCALE_DOWN_FPU`

```
#define _GLIBCXX_SCALE_DOWN_FPU
```

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `comPILEtime_settings.h`.

### 6.89.2.6 `_GLIBCXX_VERBOSE_LEVEL`

```
#define _GLIBCXX_VERBOSE_LEVEL
```

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `comPILEtime_settings.h`.

## 6.90 complex File Reference

### Classes

- struct `std::complex<_Tp>`
- struct `std::complex<_Tp>`
- struct `std::complex<double>`
- struct `std::complex<float>`
- struct `std::complex<long double>`

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define __cpp_lib_complex_udls`
- `#define _GLIBCXX_COMPLEX`

## Functions

- `template<typename _Tp >`  
`_Tp std::__complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::__complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`

- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `constexpr std::complex< double > std::literals::complex_literals::operator""i (long double __num)`
- `constexpr std::complex< double > std::literals::complex_literals::operator""i (unsigned long long __num)`
- `constexpr std::complex< float > std::literals::complex_literals::operator""if (long double __num)`
- `constexpr std::complex< float > std::literals::complex_literals::operator""if (unsigned long long __num)`
- `constexpr std::complex< long double > std::literals::complex_literals::operator""il (long double __num)`
- `constexpr std::complex< long double > std::literals::complex_literals::operator""il (unsigned long long __num)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const`  
`complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp`  
`> &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, int)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const std::complex< _Tp`  
`> &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const _Tp &__x, const`  
`std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const std::complex< _Tp`  
`> &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::proj (_Tp __x)`
- `template<typename _Tp >`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`



### 6.90.1 Detailed Description

This is a Standard C++ Library header.

## 6.91 complex File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_COMPLEX`

### Functions

- `template<typename _Tp >  
std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp, typename _Up >  
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar (const _Tp &__rho,  
const _Up &__theta)`
- `template<typename _Tp, typename _Up >  
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<  
_Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >  
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const _Tp &__x, const std::complex<  
_Up > &__y)`
- `template<typename _Tp, typename _Up >  
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<  
_Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`

### 6.91.1 Detailed Description

This is a TR1 C++ Library header.

## 6.92 complex.h File Reference

### Macros

- `#define __GLIBCXX_COMPLEX_H`

#### 6.92.1 Detailed Description

This is a Standard C++ Library header.

## 6.93 concept\_check.h File Reference

### Macros

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

#### 6.93.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.94 concurrence.h File Reference

### Classes

- class [`\_\_gnu\_cxx::\_\_scoped\_lock`](#)

### Namespaces

- [`\_\_gnu\_cxx`](#)

### Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

### Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

#### Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

#### 6.94.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 6.95 `cond_dealtor.hpp` File Reference

#### Classes

- class `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >`

#### Namespaces

- `__gnu_pbds`

#### 6.95.1 Detailed Description

Contains a conditional deallocator.

### 6.96 `cond_key_dtor_entry_dealtor.hpp` File Reference

#### Classes

- class `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >`

#### Namespaces

- `__gnu_pbds`

#### 6.96.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

### 6.97 `condition_variable` File Reference

#### Classes

- class `std::_V2::condition_variable_any`
- class `std::condition_variable`

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_CONDITION_VARIABLE`

## Enumerations

- enum [std::cv\\_status](#) { `no_timeout`, `timeout` }

## Functions

- void `std::notify_all_at_thread_exit` (`condition_variable &`, `unique_lock< mutex >`)

## 6.97.1 Detailed Description

This is a Standard C++ Library header.

6.98 `const_iterator.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_](#)< `Value_Type`, `Entry`, `Simple`, `_Alloc` >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_BIN_HEAP_CIT_BASE`

## 6.98.1 Detailed Description

Contains an iterator class returned by the table's `const` find and insert methods.

## 6.99 `const_iterator.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_< Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BASIC_HEAP_CIT_BASE`
- `#define PB_DS_CLASS_C_DEC`

#### 6.99.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 6.100 `const_iterator.hpp` File Reference

### Classes

- class [const\\_iterator\\_](#)

#### 6.100.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.

## 6.101 `constructor_destructor_fn_imps.hpp` File Reference

#### 6.101.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

## 6.102 `constructor_destructor_fn_imps.hpp` File Reference

#### 6.102.1 Detailed Description

Contains implementations of `gp_ht_map_`'s constructors, destructor, and related functions.

## 6.103 constructor\_destructor\_fn\_imps.hpp File Reference

## 6.104 constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference

### 6.104.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 6.105 constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference

### 6.105.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 6.106 constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

### 6.106.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 6.107 constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

### 6.107.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 6.108 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.108.1 Detailed Description

Contains an implementation class for binary\_heap\_.

## 6.109 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.109.1 Detailed Description

Contains an implementation for binomial\_heap\_.

## 6.110 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.110.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 6.111 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.111.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 6.112 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.112.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 6.113 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.113.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.114 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.114.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.115 constructors\_destructor\_fn\_imps.hpp File Reference

### 6.115.1 Detailed Description

Contains an implementation class for pat\_trie.

**6.116 constructors\_destructor\_fn\_imps.hpp File Reference****6.116.1 Detailed Description**

Contains an implementation for `rb_tree_`.

**6.117 constructors\_destructor\_fn\_imps.hpp File Reference****6.117.1 Detailed Description**

Contains an implementation for `rc_binomial_heap_`.

**6.118 constructors\_destructor\_fn\_imps.hpp File Reference****6.118.1 Detailed Description**

Contains an implementation class for `splay_tree_`.

**6.119 constructors\_destructor\_fn\_imps.hpp File Reference****6.119.1 Detailed Description**

Contains an implementation for `thin_heap_`.

**6.120 container\_base\_dispatch.hpp File Reference****Classes**

- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, pat\_trie\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, rb\_tree\_tag, Policy\_TI >`](#)
- [struct `\_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_TI >`](#)



## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)`
- `#define PB_DS_CHECK_KEY_EXISTS(_Key)`
- `#define PB_DS_DATA_FALSE_INDICATOR`
- `#define PB_DS_DATA_TRUE_INDICATOR`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2S(X)`
- `#define PB_DS_V2S(X)`

### 6.120.1 Detailed Description

Contains associative container dispatching.

## 6.121 `cpp_type_traits.h` File Reference

## Namespaces

- [std](#)

## Macros

- `#define __INT_N(TYPE)`

## Functions

- `template<typename _Iterator >  
_Iterator std::__miter_base (_Iterator __it)`

### 6.121.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits>`.

## 6.122 `cpu_defines.h` File Reference

### 6.122.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.123 `csetjmp` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSETJMP`
- `#define setjmp(env)`

### 6.123.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.124 `csignal` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSIGNAL`

### 6.124.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.125 cstdalign File Reference

### Macros

- `#define _GLIBCXX_CSTDALIGN`

### 6.125.1 Detailed Description

This is a Standard C++ Library header.

## 6.126 cstdarg File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSTDARG`
- `#define va_end(ap)`

### 6.126.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.127 cstdarg File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDARG`

### 6.127.1 Detailed Description

This is a TR1 C++ Library header.

## 6.128 `cstdbool` File Reference

### Macros

- `#define _GLIBCXX_CSTDBOOL`

#### 6.128.1 Detailed Description

This is a Standard C++ Library header.

## 6.129 `cstdbool` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDBOOL`

#### 6.129.1 Detailed Description

This is a TR1 C++ Library header.

## 6.130 `cstddef` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSTDDEF`

#### 6.130.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.131 `cstdint` File Reference

### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSTDINT`

##### 6.131.1 Detailed Description

This is a Standard C++ Library header.

#### 6.132 cstdint File Reference

#### Namespaces

- [std](#)
- [std::tr1](#)

#### Macros

- `#define _GLIBCXX_TR1_CSTDINT`

##### 6.132.1 Detailed Description

This is a TR1 C++ Library header.

#### 6.133 cstdio File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSTDIO`

##### 6.133.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.134 `cstdio` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDIO`

### 6.134.1 Detailed Description

This is a TR1 C++ Library header.

## 6.135 `cstdlib` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSTDLIB`
- `#define EXIT_FAILURE`
- `#define EXIT_SUCCESS`

### Functions

- void **std::abort** (void) throw ()
- int **std::atexit** (void(\*)(void)) throw ()
- void **std::exit** (int) throw ()

### 6.135.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.136 `cstdlib` File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDLIB`

#### 6.136.1 Detailed Description

This is a TR1 C++ Library header.

### 6.137 cstring File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSTRING`

#### Functions

- `void * std::memchr (void *__s, int __c, size_t __n)`
- `char * std::strchr (char *__s, int __n)`
- `char * std::strpbrk (char *__s1, const char *__s2)`
- `char * std::strrchr (char *__s, int __n)`
- `char * std::strstr (char *__s1, const char *__s2)`

#### 6.137.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 6.138 ctgmath File Reference

#### Macros

- `#define _GLIBCXX_CTGMATH`

#### 6.138.1 Detailed Description

This is a Standard C++ Library header.

## 6.139 ctgmath File Reference

### Macros

- `#define _GLIBCXX_TR1_CTGMATH`

#### 6.139.1 Detailed Description

This is a TR1 C++ Library header.

## 6.140 ctime File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CTIME`

#### 6.140.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 6.141 ctime File Reference

### Macros

- `#define _GLIBCXX_TR1_CTIME`

#### 6.141.1 Detailed Description

This is a TR1 C++ Library header.

## 6.142 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)



#### Namespaces

- [std](#)

##### 6.142.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

#### 6.143 `ctype_inline.h` File Reference

#### Namespaces

- [std](#)

##### 6.143.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

#### 6.144 `cuchar` File Reference

#### Macros

- `#define _GLIBCXX_CCHAR`

##### 6.144.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `uchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

#### 6.145 `wchar` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_WCHAR`

## Functions

- `wchar_t * std::wcschr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wcsbrk` (`wchar_t *__s1`, `const wchar_t *__s2`)
- `wchar_t * std::wcsrchr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wcsstr` (`wchar_t *__s1`, `const wchar_t *__s2`)
- `wchar_t * std::wmemchr` (`wchar_t *__p`, `wchar_t __c`, `size_t __n`)

## 6.145.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.146 `cwchar` File Reference

## Namespaces

- [std](#)
- [std::tr1](#)

## Macros

- `#define _GLIBCXX_TR1_CWCHAR`

## 6.146.1 Detailed Description

This is a TR1 C++ Library header.

6.147 `cwctype` File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_CWCTYPE`

#### 6.147.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 6.148 `cwctype` File Reference

#### Namespaces

- [std](#)
- [std::tr1](#)

#### Macros

- `#define _GLIBCXX_TR1_CWCTYPE`

#### 6.148.1 Detailed Description

This is a TR1 C++ Library header.

### 6.149 `cxxabi.h` File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)

#### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [abi](#)

#### Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type) (void *)`

## Functions

- `__cxa_dependent_exception * __cxxabiv1::__cxa_allocate_dependent_exception () noexcept`
- `int __cxxabiv1::__cxa_atexit (void(*)(void *), void *, void *) noexcept`
- `void __cxxabiv1::__cxa_bad_cast () __attribute__((__noreturn__))`
- `void __cxxabiv1::__cxa_bad_typeid () __attribute__((__noreturn__))`
- `void * __cxxabiv1::__cxa_begin_catch (void *) noexcept`
- `std::type_info * __cxxabiv1::__cxa_current_exception_type () noexcept __attribute__((__pure__))`
- `void __cxxabiv1::__cxa_deleted_virtual (void) __attribute__((__noreturn__))`
- `char * __cxxabiv1::__cxa_demangle (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)`
- `void __cxxabiv1::__cxa_end_catch ()`
- `int __cxxabiv1::__cxa_finalize (void *)`
- `void __cxxabiv1::__cxa_free_dependent_exception (__cxa_dependent_exception *) noexcept`
- `void __cxxabiv1::__cxa_free_exception (void *) noexcept`
- `void * __cxxabiv1::__cxa_get_exception_ptr (void *) noexcept __attribute__((__pure__))`
- `__cxa_eh_globals * __cxxabiv1::__cxa_get_globals () noexcept __attribute__((__const__))`
- `__cxa_eh_globals * __cxxabiv1::__cxa_get_globals_fast () noexcept __attribute__((__const__))`
- `void __cxxabiv1::__cxa_guard_abort (__guard *) noexcept`
- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *) noexcept`
- `void __cxxabiv1::__cxa_pure_virtual (void) __attribute__((__noreturn__))`
- `void __cxxabiv1::__cxa_rethrow () __attribute__((__noreturn__))`
- `int __cxxabiv1::__cxa_thread_atexit (void(*)(void *), void *, void *) noexcept`
- `void __cxxabiv1::__cxa_throw (void *, std::type_info *, void(*)(void *)) __attribute__((__noreturn__))`
- `void __cxxabiv1::__cxa_throw_bad_array_new_length () __attribute__((__noreturn__))`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void * __dest_array, void * __src_array, size_t __↵  
element_count, size_t __element_size, __cxa_cdtor_return_type(* __constructor)(void *, void *), __cxa_cdtor↵  
__type __destructor)`
- `void __cxxabiv1::__cxa_vec_cleanup (void * __array_address, size_t __element_count, size_t __s, __cxa↵  
cdtor_type __destructor) noexcept`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void * __array_address, size_t __element_count,  
size_t __element_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_delete (void * __array_address, size_t __element_size, size_t __padding_size,  
__cxa_cdtor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_delete2 (void * __array_address, size_t __element_size, size_t __padding_size,  
__cxa_cdtor_type __destructor, void(* __dealloc)(void *))`
- `void __cxxabiv1::__cxa_vec_delete3 (void * __array_address, size_t __element_size, size_t __padding_size,  
__cxa_cdtor_type __destructor, void(* __dealloc)(void *, size_t))`
- `void __cxxabiv1::__cxa_vec_dtor (void * __array_address, size_t __element_count, size_t __element_size, ↵  
__cxa_cdtor_type __destructor)`
- `void * __cxxabiv1::__cxa_vec_new (size_t __element_count, size_t __element_size, size_t __padding_size,  
__cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)`
- `void * __cxxabiv1::__cxa_vec_new2 (size_t __element_count, size_t __element_size, size_t __padding_size,  
__cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void(* __alloc)(size_t), void(* __dealloc)(void  
*))`
- `void * __cxxabiv1::__cxa_vec_new3 (size_t __element_count, size_t __element_size, size_t __padding_size,  
__cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void(* __alloc)(size_t), void(* __dealloc)(void  
*, size_t))`
- `void * __cxxabiv1::__dynamic_cast (const void * __src_ptr, const __class_type_info * __src_type, const __↵  
class_type_info * __dst_type, ptrdiff_t __src2dst)`

### 6.149.1 Detailed Description

The header provides an interface to the C++ ABI.

### 6.149.2 Function Documentation

#### 6.149.2.1 `__cxa_demangle()`

```
char* __cxxabiv1::__cxa_demangle (
    const char * __mangled_name,
    char * __output_buffer,
    size_t * __length,
    int * __status )
```

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

#### Parameters

<code>__mangled_name</code>	A NUL-terminated character string containing the name to be demangled.
<code>__output_buffer</code>	A region of memory, allocated with <code>malloc</code> , of <code>*__length</code> bytes, into which the demangled name is stored. If <code>__output_buffer</code> is not long enough, it is expanded using <code>realloc</code> . <code>__output_buffer</code> may instead be <code>NULL</code> ; in that case, the demangled name is placed in a region of memory allocated with <code>malloc</code> .
<code>__length</code>	If <code>__length</code> is non- <code>NULL</code> , the length of the buffer containing the demangled name is placed in <code>*__length</code> .
<code>__status</code>	<code>*__status</code> is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: <code>mangled_name</code> is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.

#### Returns

A pointer to the start of the NUL-terminated demangled name, or `NULL` if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in `__gnu_cxx::__verbose_terminate_handler`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_demangling.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_demangling.html) for other examples of use.

#### Note

The same demangling functionality is available via `libiberty` (`<libiberty/demangle.h>` and `libiberty`.↵a) in GCC 3.1 and later, but that requires explicit installation (`-enable-install-libiberty`) and uses a different API, although the ABI is unchanged.

## 6.150 `cxxabi_forced.h` File Reference

### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)

#### 6.150.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 6.151 `cxxabi_init_exception.h` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CDTOR_CALLABI`
- `#define _GLIBCXX_HAVE_CDTOR_CALLABI`

### Functions

- `void * __cxxabiv1::__cxa_allocate_exception (size_t) noexcept`
- `void __cxxabiv1::__cxa_free_exception (void *) noexcept`
- `__cxa_refcounted_exception * __cxxabiv1::__cxa_init_primary_exception (void *object, std::type\_info *tinfo, void(*dest)(void *)) noexcept`

#### 6.151.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.152 `cxxabi_tweaks.h` File Reference

### Macros

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

## Typedefs

- typedef void **\_\_cxxabiv1::\_\_cxa\_cdtor\_return\_type**
- typedef void **\_\_cxxabiv1::\_\_cxa\_vec\_ctor\_return\_type**

## Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI__)))`

### 6.152.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 6.153 debug.h File Reference

### Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive(_First, _Last)`
- `#define __glibcxx_requires_irreflexive2(_First, _Last)`
- `#define __glibcxx_requires_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`

### 6.153.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.154 debug\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator<\\_Alloc>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Alloc>  
bool [\\_\\_gnu\\_cxx::operator!=](#) (const debug\_allocator<\_Alloc> &\_\_lhs, const debug\_allocator<\_Alloc> &\_\_rhs)

#### 6.154.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.155 debug\_fn\_imps.hpp File Reference

#### 6.155.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 6.156 debug\_fn\_imps.hpp File Reference

#### 6.156.1 Detailed Description

Contains an implementation for binomial\_heap\_.

## 6.157 debug\_fn\_imps.hpp File Reference

#### 6.157.1 Detailed Description

Contains an implementation class for a base of binomial heaps.



## 6.158 `debug_fn_imps.hpp` File Reference

### 6.158.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.159 `debug_fn_imps.hpp` File Reference

### 6.159.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

## 6.160 `debug_fn_imps.hpp` File Reference

### 6.160.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

## 6.161 `debug_fn_imps.hpp` File Reference

### 6.161.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.162 `debug_fn_imps.hpp` File Reference

### 6.162.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

## 6.163 `debug_fn_imps.hpp` File Reference

### 6.163.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 6.164 `debug_fn_imps.hpp` File Reference

### 6.164.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.165 `debug_fn_imps.hpp` File Reference

### 6.165.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 6.166 `debug_fn_imps.hpp` File Reference

### 6.166.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 6.167 `debug_fn_imps.hpp` File Reference

### 6.167.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 6.168 `debug_fn_imps.hpp` File Reference

### 6.168.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 6.169 `debug_fn_imps.hpp` File Reference

### 6.169.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 6.170 `debug_map_base.hpp` File Reference

### 6.170.1 Detailed Description

Contains a debug-mode base for all maps.

## 6.171 `debug_no_store_hash_fn_imps.hpp` File Reference

### 6.171.1 Detailed Description

Contains implementations of `cc_ht_map_'s` debug-mode functions.

## 6.172 `debug_no_store_hash_fn_imps.hpp` File Reference

### 6.172.1 Detailed Description

Contains implementations of `gp_ht_map_'s` debug-mode functions.

## 6.173 `debug_store_hash_fn_imps.hpp` File Reference

### 6.173.1 Detailed Description

Contains implementations of `cc_ht_map_'s` debug-mode functions.

## 6.174 `debug_store_hash_fn_imps.hpp` File Reference

### 6.174.1 Detailed Description

Contains implementations of `gp_ht_map_'s` debug-mode functions.

## 6.175 `decimal` File Reference

### Classes

- class [std::decimal::decimal128](#)
- class [std::decimal::decimal32](#)
- class [std::decimal::decimal64](#)

## Namespaces

- [std](#)
- [std::decimal](#)

## Macros

- `#define DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT(_Op)`
- `#define DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT(_Op)`
- `#define DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT(_Op)`
- `#define DECLARE_DECIMAL_BINARY_OP_WITH_DEC(_Op, _T1, _T2, _T3)`
- `#define DECLARE_DECIMAL_BINARY_OP_WITH_INT(_Op, _Tp)`
- `#define DECLARE_DECIMAL_COMPARISON(_Op, _Tp)`
- `#define GLIBCXX_DECIMAL`
- `#define` `GLIBCXX_USE_DECIMAL`

## Functions

- double `std::decimal::decimal128_to_double` (decimal128 \_\_d)
- float `std::decimal::decimal128_to_float` (decimal128 \_\_d)
- long double `std::decimal::decimal128_to_long_double` (decimal128 \_\_d)
- long long `std::decimal::decimal128_to_long_long` (decimal128 \_\_d)
- double `std::decimal::decimal32_to_double` (decimal32 \_\_d)
- float `std::decimal::decimal32_to_float` (decimal32 \_\_d)
- long double `std::decimal::decimal32_to_long_double` (decimal32 \_\_d)
- long long `std::decimal::decimal32_to_long_long` (decimal32 \_\_d)
- double `std::decimal::decimal64_to_double` (decimal64 \_\_d)
- float `std::decimal::decimal64_to_float` (decimal64 \_\_d)
- long double `std::decimal::decimal64_to_long_double` (decimal64 \_\_d)
- long long `std::decimal::decimal64_to_long_long` (decimal64 \_\_d)
- double `std::decimal::decimal_to_double` (decimal32 \_\_d)
- double `std::decimal::decimal_to_double` (decimal64 \_\_d)
- double `std::decimal::decimal_to_double` (decimal128 \_\_d)
- float `std::decimal::decimal_to_float` (decimal32 \_\_d)
- float `std::decimal::decimal_to_float` (decimal64 \_\_d)
- float `std::decimal::decimal_to_float` (decimal128 \_\_d)
- long double `std::decimal::decimal_to_long_double` (decimal32 \_\_d)
- long double `std::decimal::decimal_to_long_double` (decimal64 \_\_d)
- long double `std::decimal::decimal_to_long_double` (decimal128 \_\_d)
- long long `std::decimal::decimal_to_long_long` (decimal32 \_\_d)
- long long `std::decimal::decimal_to_long_long` (decimal64 \_\_d)
- long long `std::decimal::decimal_to_long_long` (decimal128 \_\_d)
- static decimal128 `std::decimal::make_decimal128` (long long \_\_coeff, int \_\_exp)
- static decimal128 `std::decimal::make_decimal128` (unsigned long long \_\_coeff, int \_\_exp)
- static decimal32 `std::decimal::make_decimal32` (long long \_\_coeff, int \_\_exp)
- static decimal32 `std::decimal::make_decimal32` (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 `std::decimal::make_decimal64` (long long \_\_coeff, int \_\_exp)
- static decimal64 `std::decimal::make_decimal64` (unsigned long long \_\_coeff, int \_\_exp)
- bool `std::decimal::operator!=` (decimal32 \_\_lhs, decimal32 \_\_rhs)

- Generated by Doxygen

- decimal32 **std::decimal::operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)

- Generated by Doxygen

- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)



- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal64 \_\_rhs)

- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal128 \_\_rhs)

- [illegible]

- `bool std::decimal::operator> (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator> (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator> (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator> (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator> (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator> (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator> (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal128 __rhs)`

### 6.175.1 Detailed Description

This is a Standard C++ Library header.

## 6.176 deque File Reference

### Macros

- `#define _GLIBCXX_DEQUE`

### 6.176.1 Detailed Description

This is a Standard C++ Library header.

## 6.177 deque File Reference

### Classes

- class `std::__debug::deque<_Tp, _Allocator>`

### Namespaces

- `std`
- `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_DEQUE`

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const deque<_Tp, _Alloc> &__lhs, const deque<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (deque<_Tp, _Alloc> &__lhs, deque<_Tp, _Alloc> &__rhs) noexcept(*conditional *)`

## 6.177.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.178 deque File Reference

## Classes

- class [std::\\_\\_profile::deque<\\_Tp, \\_Allocator >](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Macros

- `#define _GLIBCXX_PROFILE_DEQUE`

## Functions

- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::__profile::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(*conditional  
*/)`

## 6.178.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.179 deque File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_DEQUE`

## Typedefs

- `template<typename _Tp >`  
using **`std::experimental::fundamentals_v2::pmr::deque`** = `std::deque`< \_Tp, polymorphic\_allocator< \_Tp >  
>

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >`  
void **`std::experimental::fundamentals_v2::erase`** (deque< \_Tp, \_Alloc > &\_\_cont, const \_Up &\_\_value)
- `template<typename _Tp, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (deque< \_Tp, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

### 6.179.1 Detailed Description

This is a TS C++ Library header.

## 6.180 deque.tcc File Reference

### Namespaces

- `std`

## Macros

- `#define _DEQUE_TCC`

## Functions

- `template<typename _Tp >`  
`_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > **`std::copy`** (`_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp \* > \_\_first,  
`_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp \* > \_\_last, `_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > \_\_result)
- `template<typename _Tp >`  
`_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > **`std::copy_backward`** (`_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp  
\* > \_\_first, `_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp \* > \_\_last, `_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* >  
\_\_result)
- `template<typename _Tp >`  
void **`std::fill`** (const `_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > &\_\_first, const `_Deque_iterator`< \_Tp, \_Tp &, \_Tp \*  
> &\_\_last, const \_Tp &\_\_value)
- `template<typename _Tp >`  
`_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > **`std::move`** (`_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp \* > \_\_first,  
`_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp \* > \_\_last, `_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > \_\_result)
- `template<typename _Tp >`  
`_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* > **`std::move_backward`** (`_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp  
\* > \_\_first, `_Deque_iterator`< \_Tp, const \_Tp &, const \_Tp \* > \_\_last, `_Deque_iterator`< \_Tp, \_Tp &, \_Tp \* >  
\_\_result)

## 6.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

6.181 `direct_mask_range_hashing_imp.hpp` File Reference

## 6.181.1 Detailed Description

Contains a range-hashing policy implementation

6.182 `direct_mod_range_hashing_imp.hpp` File Reference

## 6.182.1 Detailed Description

Contains a range-hashing policy implementation

6.183 `dynamic_bitset` File Reference

## Classes

- struct `std::tr2::__dynamic_bitset_base<_WordT, _Alloc>`
- class `std::tr2::dynamic_bitset<_WordT, _Alloc>`
- class `std::tr2::dynamic_bitset<_WordT, _Alloc>::reference`

## Namespaces

- `std`
- `std::tr2`

## Macros

- `#define _GLIBCXX_TR2_DYNAMIC_BITSET`



## Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const dynamic_bitset< _WordT, _Alloc > &__x)`
- `template<typename _WordT, typename _Alloc >`  
`bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc`  
`> &__rhs)`
- `template<typename _WordT, typename _Alloc >`  
`bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc`  
`> &__rhs)`
- `template<typename _WordT, typename _Alloc >`  
`bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc`  
`> &__rhs)`
- `template<typename _WordT, typename _Alloc >`  
`bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc`  
`> &__rhs)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`

### 6.183.1 Detailed Description

This is a TR2 C++ Library header.

## 6.184 dynamic\_bitset.tcc File Reference

### Namespaces

- `std`
- `std::tr2`

## Macros

- `#define _GLIBCXX_TR2_DYNAMIC_BITSET_TCC`

## Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >  
std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__x  
is, dynamic_bitset< _WordT, _Alloc > &__x)`

### 6.184.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<tr2/dynamic_bitset>`.

## 6.185 enable\_special\_members.h File Reference

### Classes

- `struct std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`
- `struct std::_Enable_default_constructor< _Switch, _Tag >`
- `struct std::_Enable_destructor< _Switch, _Tag >`
- `struct std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`

### Namespaces

- `std`

### 6.185.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.186 enc\_filebuf.h File Reference

### Classes

- `class __gnu_cxx::enc_filebuf< _CharT >`

### Namespaces

- `__gnu_cxx`

#### 6.186.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 6.187 `entry_cmp.hpp` File Reference

#### Classes

- [struct `\_\_gnu\_pbds::detail::entry\_cmp<\_VTp, Cmp\_Fn, \_Alloc, No\_Throw >`](#)
- [struct `\_\_gnu\_pbds::detail::entry\_cmp<\_VTp, Cmp\_Fn, \_Alloc, false >`](#)
- [struct `\_\_gnu\_pbds::detail::entry\_cmp<\_VTp, Cmp\_Fn, \_Alloc, false >::type`](#)
- [struct `\_\_gnu\_pbds::detail::entry\_cmp<\_VTp, Cmp\_Fn, \_Alloc, true >`](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.187.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.188 `entry_list_fn_imps.hpp` File Reference

#### 6.188.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entry-list related functions.

### 6.189 `entry_metadata_base.hpp` File Reference

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.189.1 Detailed Description

Contains an implementation for a list update map.

## 6.190 entry\_pred.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, No\\_Throw >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.190.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 6.191 eq\_by\_less.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::eq\\_by\\_less< Key, Cmp\\_Fn >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.191.1 Detailed Description

Contains an equivalence function.

## 6.192 equally\_split.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_DifferenceType , typename \_OutputIterator >  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_equally\\_split](#) (\_DifferenceType \_\_n, \_ThreadIndex \_\_num\_threads, \_OutputIterator \_\_s)
- template<typename \_DifferenceType >  
\_DifferenceType [\\_\\_gnu\\_parallel::\\_\\_equally\\_split\\_point](#) (\_DifferenceType \_\_n, \_ThreadIndex \_\_num\_threads, \_ThreadIndex \_\_thread\_no)

#### 6.192.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

### 6.193 `erase_fn_imps.hpp` File Reference

#### 6.193.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.194 `erase_fn_imps.hpp` File Reference

#### 6.194.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

### 6.195 `erase_fn_imps.hpp` File Reference

#### 6.195.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 6.196 `erase_fn_imps.hpp` File Reference

#### 6.196.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions.

### 6.197 `erase_fn_imps.hpp` File Reference

#### 6.197.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions.

### 6.198 `erase_fn_imps.hpp` File Reference

#### 6.198.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.199 erase\_fn\_imps.hpp File Reference

### 6.199.1 Detailed Description

Contains implementations of lu\_map\_.

## 6.200 erase\_fn\_imps.hpp File Reference

### 6.200.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.201 erase\_fn\_imps.hpp File Reference

### 6.201.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.202 erase\_fn\_imps.hpp File Reference

### 6.202.1 Detailed Description

Contains an implementation class for pat\_trie.

## 6.203 erase\_fn\_imps.hpp File Reference

### 6.203.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.204 erase\_fn\_imps.hpp File Reference

### 6.204.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 6.205 `erase_fn_imps.hpp` File Reference

### 6.205.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 6.206 `erase_fn_imps.hpp` File Reference

### 6.206.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 6.207 `erase_if.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _Container, typename _Predicate >  
void std::experimental::fundamentals_v2::__detail::__erase_nodes_if (_Container &__cont, _Predicate _↔  
_pred)`

### 6.207.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.208 `erase_no_store_hash_fn_imps.hpp` File Reference

### 6.208.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is not stored.

## 6.209 `erase_no_store_hash_fn_imps.hpp` File Reference

### 6.209.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is not stored.

## 6.210 erase\_store\_hash\_fn\_imps.hpp File Reference

### 6.210.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s erase related functions, when the hash value is stored.

## 6.211 erase\_store\_hash\_fn\_imps.hpp File Reference

### 6.211.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s erase related functions, when the hash value is stored.

## 6.212 error\_constants.h File Reference

### Namespaces

- [std](#)

### Enumerations

- `enum errc {  
    address_family_not_supported, address_in_use, address_not_available, already_connected,  
    argument_list_too_long, argument_out_of_domain, bad_address, bad_file_descriptor,  
    broken_pipe, connection_aborted, connection_already_in_progress, connection_refused,  
    connection_reset, cross_device_link, destination_address_required, device_or_resource_busy,  
    directory_not_empty, executable_format_error, file_exists, file_too_large,  
    filename_too_long, function_not_supported, host_unreachable, illegal_byte_sequence,  
    inappropriate_io_control_operation, interrupted, invalid_argument, invalid_seek,  
    io_error, is_a_directory, message_size, network_down,  
    network_reset, network_unreachable, no_buffer_space, no_child_process,  
    no_lock_available, no_message, no_protocol_option, no_space_on_device,  
    no_such_device_or_address, no_such_device, no_such_file_or_directory, no_such_process,  
    not_a_directory, not_a_socket, not_connected, not_enough_memory,  
    operation_in_progress, operation_not_permitted, operation_not_supported, operation_would_block,  
    permission_denied, protocol_not_supported, read_only_file_system, resource_deadlock_would_occur,  
    resource_unavailable_try_again, result_out_of_range, timed_out, too_many_files_open_in_system,  
    too_many_files_open, too_many_links, too_many_symbolic_link_levels, wrong_protocol_type }`

### 6.212.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.



## 6.213 exception File Reference

### Classes

- class [std::bad\\_exception](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define __cpp_lib_uncaught_exceptions`
- `#define` [\\_\\_EXCEPTION\\_\\_](#)

### Typedefs

- typedef void(\* [std::terminate\\_handler](#)) ()
- typedef void(\* [std::unexpected\\_handler](#)) ()

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- terminate\_handler [std::get\\_terminate](#) () noexcept
- unexpected\_handler [std::get\\_unexpected](#) () noexcept
- terminate\_handler [std::set\\_terminate](#) (terminate\_handler) noexcept
- unexpected\_handler [std::set\\_unexpected](#) (unexpected\_handler) noexcept
- void [std::terminate](#) () noexcept `__attribute__((__noreturn__))`
- `_GLIBCXX17_DEPRECATED` bool [std::uncaught\\_exception](#) () noexcept `__attribute__((__pure__))`
- int [std::uncaught\\_exceptions](#) () noexcept `__attribute__((__pure__))`
- void [std::unexpected](#) () `__attribute__((__noreturn__))`

### 6.213.1 Detailed Description

This is a Standard C++ Library header.

## 6.214 exception.h File Reference

### Classes

- class [std::exception](#)

## Namespaces

- [std](#)

## 6.214.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.215 exception.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)
- struct [\\_\\_gnu\\_pbds::insert\\_error](#)
- struct [\\_\\_gnu\\_pbds::join\\_error](#)
- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error\(\)](#)

## 6.215.1 Detailed Description

Contains exception classes.

## 6.216 exception\_defines.h File Reference

## Macros

- `#define \_\_catch\(X\)`
- `#define \_\_throw\_exception\_again`
- `#define \_\_try`

### 6.216.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 6.217 `exception_ptr.h` File Reference

### Classes

- class `std::__exception_ptr::exception_ptr`

### Namespaces

- `std`

### Functions

- `template<typename _Ex >`  
`void std::__exception_ptr::__dest_thunk (void *__x)`
- `exception_ptr std::current_exception () noexcept`
- `template<typename _Ex >`  
`exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `bool std::__exception_ptr::operator!= (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `bool std::__exception_ptr::operator== (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::__exception_ptr::swap (exception_ptr &__lhs, exception_ptr &__rhs)`

### 6.217.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 6.218 `extc++.h` File Reference

### 6.218.1 Detailed Description

This is an implementation file for a precompiled header.

## 6.219 `extptr_allocator.h` File Reference

### Classes

- class `__gnu_cxx::ExtPtr_allocator<_Tp>`

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp >  
void __gnu_cxx::swap (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`

### 6.219.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

## 6.220 features.h File Reference

## Macros

- `#define \_GLIBCXX\_BAL\_QUICKSORT`
- `#define \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS`
- `#define \_GLIBCXX\_FIND\_EQUAL\_SPLIT`
- `#define \_GLIBCXX\_FIND\_GROWING\_BLOCKS`
- `#define \_GLIBCXX\_MERGESORT`
- `#define \_GLIBCXX\_QUICKSORT`
- `#define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING`
- `#define \_GLIBCXX\_TREE\_FULL\_COPY`
- `#define \_GLIBCXX\_TREE\_INITIAL\_SPLITTING`

### 6.220.1 Detailed Description

Defines on whether to include algorithm variants.

Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

### 6.220.2 Macro Definition Documentation

#### 6.220.2.1 `_GLIBCXX_BAL_QUICKSORT`

```
#define _GLIBCXX_BAL_QUICKSORT
```

Include parallel dynamically load-balanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file features.h.

#### 6.220.2.2 `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

```
#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
```

Include the equal-sized blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file features.h.

#### 6.220.2.3 `_GLIBCXX_FIND_EQUAL_SPLIT`

```
#define _GLIBCXX_FIND_EQUAL_SPLIT
```

Include the equal splitting variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file features.h.

#### 6.220.2.4 `_GLIBCXX_FIND_GROWING_BLOCKS`

```
#define _GLIBCXX_FIND_GROWING_BLOCKS
```

Include the growing blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file features.h.

#### 6.220.2.5 \_GLIBCXX\_MERGESORT

```
#define _GLIBCXX_MERGESORT
```

Include parallel multi-way mergesort.

See also

`__gnu_parallel::Settings::sort_algorithm`

Definition at line 41 of file features.h.

#### 6.220.2.6 \_GLIBCXX\_QUICKSORT

```
#define _GLIBCXX_QUICKSORT
```

Include parallel unbalanced quicksort.

See also

`__gnu_parallel::Settings::sort_algorithm`

Definition at line 48 of file features.h.

#### 6.220.2.7 \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING

```
#define _GLIBCXX_TREE_DYNAMIC_BALANCING
```

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file features.h.

#### 6.220.2.8 \_GLIBCXX\_TREE\_FULL\_COPY

```
#define _GLIBCXX_TREE_FULL_COPY
```

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file features.h.

### 6.220.2.9 `_GLIBCXX_TREE_INITIAL_SPLITTING`

```
#define _GLIBCXX_TREE_INITIAL_SPLITTING
```

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

## 6.221 `fenv.h` File Reference

### 6.221.1 Detailed Description

This is a Standard C++ Library header.

## 6.222 `filesystem` File Reference

Macros

- `#define __cpp_lib_experimental_filesystem`
- `#define _GLIBCXX_EXPERIMENTAL_FILESYSTEM`

### 6.222.1 Detailed Description

This is a TS C++ Library header.

## 6.223 `find.h` File Reference

Namespaces

- [`\_\_gnu\_parallel`](#)

Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`

#### 6.223.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

### 6.224 find\_fn\_imps.hpp File Reference

#### 6.224.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.225 find\_fn\_imps.hpp File Reference

#### 6.225.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

### 6.226 find\_fn\_imps.hpp File Reference

#### 6.226.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 6.227 find\_fn\_imps.hpp File Reference

#### 6.227.1 Detailed Description

Contains implementations of `cc_ht_map_`'s find related functions.

### 6.228 find\_fn\_imps.hpp File Reference

#### 6.228.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions.

### 6.229 find\_fn\_imps.hpp File Reference

#### 6.229.1 Detailed Description

Contains implementations of `lu_map_`.



## 6.230 `find_fn_imps.hpp` File Reference

### 6.230.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.231 `find_fn_imps.hpp` File Reference

### 6.231.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.232 `find_fn_imps.hpp` File Reference

### 6.232.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 6.233 `find_fn_imps.hpp` File Reference

### 6.233.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 6.234 `find_fn_imps.hpp` File Reference

### 6.234.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 6.235 `find_no_store_hash_fn_imps.hpp` File Reference

### 6.235.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions, when the hash value is not stored.

## 6.236 find\_selectors.h File Reference

## Classes

- [struct \\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## 6.236.1 Detailed Description

\_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

## 6.237 find\_store\_hash\_fn\_imps.hpp File Reference

## 6.237.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s find related functions, when the hash value is stored.

## 6.238 find\_store\_hash\_fn\_imps.hpp File Reference

## 6.238.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s insert related functions, when the hash value is stored.

## 6.239 for\_each.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- [template<typename \\_Iter, typename \\_UserOp, typename \\_Functionality, typename \\_Red, typename \\_Result> \\_UserOp \\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\(\\_Iter \\_\\_begin, \\_Iter \\_\\_end, \\_UserOp \\_\\_user←\\_op, \\_Functionality & \\_\\_functionality, \\_Red \\_\\_reduction, \\_Result \\_\\_reduction\\_start, \\_Result & \\_\\_output, typename std::iterator\\_traits<\\_Iter>::difference\\_type \\_\\_bound, \\_Parallelism \\_\\_parallelism\\_tag\)](#)

### 6.239.1 Detailed Description

Main interface for embarrassingly parallel functions.

The explicit implementation are in other header files, like `workstealing.h`, `par_loop.h`, `omp_loop.h`, and `omp_loop_↵static.h`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.240 `for_each_selectors.h` File Reference

### Classes

- [struct `\_\_gnu\_parallel::\_\_accumulate\_binop\_reduct<\_BinOp>`](#)
- [struct `\_\_gnu\_parallel::\_\_accumulate\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_adjacent\_difference\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_count\_if\_selector<\_It, \_Diff>`](#)
- [struct `\_\_gnu\_parallel::\_\_count\_selector<\_It, \_Diff>`](#)
- [struct `\_\_gnu\_parallel::\_\_fill\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_for\_each\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_generate\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_identity\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_inner\_product\_selector<\_It, \_It2, \_Tp>`](#)
- [struct `\_\_gnu\_parallel::\_\_max\_element\_reduct<\_Compare, \_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_min\_element\_reduct<\_Compare, \_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_replace\_if\_selector<\_It, \_Op, \_Tp>`](#)
- [struct `\_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>`](#)
- [struct `\_\_gnu\_parallel::\_\_transform1\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_transform2\_selector<\_It>`](#)
- [struct `\_\_gnu\_parallel::\_\_DummyReduct`](#)
- [struct `\_\_gnu\_parallel::\_\_Nothing`](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### 6.240.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

## 6.241 `formatter.h` File Reference

### Classes

- [class `\_\_gnu\_debug::\_\_Safe\_iterator<\_Iterator, \_Sequence>`](#)
- [class `\_\_gnu\_debug::\_\_Safe\_local\_iterator<\_Iterator, \_Sequence>`](#)
- [class `\_\_gnu\_debug::\_\_Safe\_sequence<\_Sequence>`](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Macros

- `#define _GLIBCXX_TYPEID(_Type)`

## Enumerations

- `enum _Debug_msg_id {`  
`__msg_valid_range, __msg_insert_singular, __msg_insert_different, __msg_erase_bad,`  
`__msg_erase_different, __msg_subscript_oob, __msg_empty, __msg_unpartitioned,`  
`__msg_unpartitioned_pred, __msg_unsorted, __msg_unsorted_pred, __msg_not_heap,`  
`__msg_not_heap_pred, __msg_bad_bitset_write, __msg_bad_bitset_read, __msg_bad_bitset_flip,`  
`__msg_self_splice, __msg_splice_alloc, __msg_splice_bad, __msg_splice_other,`  
`__msg_splice_overlap, __msg_init_singular, __msg_init_copy_singular, __msg_init_const_singular,`  
`__msg_copy_singular, __msg_bad_deref, __msg_bad_inc, __msg_bad_dec,`  
`__msg_iter_subscript_oob, __msg_advance_oob, __msg_retreat_oob, __msg_iter_compare_bad,`  
`__msg_compare_different, __msg_iter_order_bad, __msg_order_different, __msg_distance_bad,`  
`__msg_distance_different, __msg_deref_istream, __msg_inc_istream, __msg_output_ostream,`  
`__msg_deref_istreambuf, __msg_inc_istreambuf, __msg_insert_after_end, __msg_erase_after_bad,`  
`__msg_valid_range2, __msg_local_iter_compare_bad, __msg_non_empty_range, __msg_self_move_↵`  
`assign,`  
`__msg_bucket_index_oob, __msg_valid_load_factor, __msg_equal_allocs, __msg_insert_range_from_↵`  
`__self,`  
`__msg_irreflexive_ordering }`

## Functions

- `template<typename _Iterator >`  
`bool __gnu_debug::__check_singular (const _Iterator &)`

## 6.241.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.242 forward\_list File Reference

## Macros

- `#define _GLIBCXX_FORWARD_LIST`

## 6.242.1 Detailed Description

This is a Standard C++ Library header.

## 6.243 forward\_list File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_forward\\_list<\\_SafeSequence >](#)
- class [std::\\_\\_debug::forward\\_list<\\_Tp, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define \_\_glibcxx\_check\_valid\_fl\_range(_First, _Last, _Dist)`
- `#define GLIBCXX\_DEBUG\_FORWARD\_LIST`

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
void std::\_\_debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`

#### 6.243.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.244 forward\_list File Reference

## Classes

- class [std::\\_\\_profile::forward\\_list<\\_Tp, \\_Alloc>](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Macros

- `#define GLIBCXX_PROFILE_FORWARD_LIST`

## Functions

- `template<typename _Tp, typename _Alloc>  
bool std::\_\_profile::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
bool std::\_\_profile::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
bool std::\_\_profile::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
bool std::\_\_profile::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
bool std::\_\_profile::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
bool std::\_\_profile::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
void std::\_\_profile::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`

## 6.244.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.245 forward\_list File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_FORWARD_LIST`

## Typedefs

- `template<typename _Tp >  
using std::experimental::fundamentals_v2::pmr::forward_list = std::forward\_list< _Tp, polymorphic\_allocator< _Tp > >`

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >  
void std::experimental::fundamentals_v2::erase (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate >  
void std::experimental::fundamentals_v2::erase_if (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 6.245.1 Detailed Description

This is a TS C++ Library header.

### 6.246 `forward_list.h` File Reference

## Classes

- struct [std::\\_Fwd\\_list\\_base](#)< \_Tp, \_Alloc >
- struct [std::\\_Fwd\\_list\\_const\\_iterator](#)< \_Tp >
- struct [std::\\_Fwd\\_list\\_iterator](#)< \_Tp >
- struct [std::\\_Fwd\\_list\\_node](#)< \_Tp >
- struct [std::\\_Fwd\\_list\\_node\\_base](#)
- class [std::forward\\_list](#)< \_Tp, \_Alloc >

## Namespaces

- [std](#)

## Functions

- `template<typename _Tp >`  
`bool std::operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp >`  
`bool std::operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`

## 6.246.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 6.247 forward\_list.tcc File Reference

## Namespaces

- `std`

## Macros

- `#define _FORWARD_LIST_TCC`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

## 6.247.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.



## 6.248 fs\_dir.h File Reference

### 6.248.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.249 fs\_dir.h File Reference

### Namespaces

- [std](#)

### Functions

- `directory_iterator` **std::experimental::filesystem::v1::begin** (`directory_iterator __iter`) `noexcept`
- `directory_iterator` **std::experimental::filesystem::v1::end** (`directory_iterator`) `noexcept`
- `bool` **std::experimental::filesystem::v1::operator==** (`const directory_iterator &__lhs`, `const directory_iterator &__rhs`)
- `bool` **std::experimental::filesystem::v1::operator!=** (`const directory_iterator &__lhs`, `const directory_iterator &__rhs`)
- `recursive_directory_iterator` **std::experimental::filesystem::v1::begin** (`recursive_directory_iterator __iter`) `noexcept`
- `recursive_directory_iterator` **std::experimental::filesystem::v1::end** (`recursive_directory_iterator`) `noexcept`
- `bool` **std::experimental::filesystem::v1::operator==** (`const recursive_directory_iterator &__lhs`, `const recursive_directory_iterator &__rhs`)
- `bool` **std::experimental::filesystem::v1::operator!=** (`const recursive_directory_iterator &__lhs`, `const recursive_directory_iterator &__rhs`)

### 6.249.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.250 fs\_fwd.h File Reference

### 6.250.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.251 fs\_fwd.h File Reference

## Namespaces

- [std](#)

## Typedefs

- using **std::experimental::filesystem::v1::file\_time\_type** = std::chrono::system\_clock::time\_point

## Enumerations

- enum [std::experimental::filesystem::v1::copy\\_options](#) : unsigned short { **none**, **skip\_existing**, **overwrite\_existing**, **update\_existing**, **recursive**, **copy\_symlinks**, **skip\_symlinks**, **directories\_only**, **create\_symlinks**, **create\_hard\_links** }
- enum **directory\_options** : unsigned char { **none**, **follow\_directory\_symlink**, **skip\_permission\_denied** }
- enum **file\_type** : signed char { **none**, **not\_found**, **regular**, **directory**, **symlink**, **block**, **character**, **fifo**, **socket**, **unknown** }
- enum [std::experimental::filesystem::v1::perms](#) : unsigned { **none**, **owner\_read**, **owner\_write**, **owner\_exec**, **owner\_all**, **group\_read**, **group\_write**, **group\_exec**, **group\_all**, **others\_read**, **others\_write**, **others\_exec**, **others\_all**, **all**, **set\_uid**, **set\_gid**, **sticky\_bit**, **mask**, **unknown**, **add\_perms**, **remove\_perms**, **symlink\_nofollow** }

## Functions

- constexpr copy\_options **std::experimental::filesystem::v1::operator&** (copy\_options \_\_x, copy\_options \_\_y) noexcept
- constexpr perms **std::experimental::filesystem::v1::operator&** (perms \_\_x, perms \_\_y) noexcept
- constexpr directory\_options **std::experimental::filesystem::v1::operator&** (directory\_options \_\_x, directory\_options \_\_y) noexcept
- copy\_options & **std::experimental::filesystem::v1::operator&=** (copy\_options &\_\_x, copy\_options \_\_y) noexcept
- perms & **std::experimental::filesystem::v1::operator&=** (perms &\_\_x, perms \_\_y) noexcept
- directory\_options & **std::experimental::filesystem::v1::operator&=** (directory\_options &\_\_x, directory\_options \_\_y) noexcept
- constexpr copy\_options **std::experimental::filesystem::v1::operator^** (copy\_options \_\_x, copy\_options \_\_y) noexcept
- constexpr perms **std::experimental::filesystem::v1::operator^** (perms \_\_x, perms \_\_y) noexcept
- constexpr directory\_options **std::experimental::filesystem::v1::operator^** (directory\_options \_\_x, directory\_options \_\_y) noexcept
- copy\_options & **std::experimental::filesystem::v1::operator^=** (copy\_options &\_\_x, copy\_options \_\_y) noexcept
- perms & **std::experimental::filesystem::v1::operator^=** (perms &\_\_x, perms \_\_y) noexcept

- `directory_options & std::experimental::filesystem::v1::operator^= (directory_options __x, directory_options __y) noexcept`
  - `constexpr copy_options std::experimental::filesystem::v1::operator| (copy_options __x, copy_options __y) noexcept`
  - `constexpr perms std::experimental::filesystem::v1::operator| (perms __x, perms __y) noexcept`
  - `constexpr directory_options std::experimental::filesystem::v1::operator| (directory_options __x, directory_options __y) noexcept`
  - `copy_options & std::experimental::filesystem::v1::operator|= (copy_options __x, copy_options __y) noexcept`
  - `perms & std::experimental::filesystem::v1::operator|= (perms __x, perms __y) noexcept`
  - `directory_options & std::experimental::filesystem::v1::operator|= (directory_options __x, directory_options __y) noexcept`
  - `constexpr copy_options std::experimental::filesystem::v1::operator~ (copy_options __x) noexcept`
  - `constexpr perms std::experimental::filesystem::v1::operator~ (perms __x) noexcept`
  - `constexpr directory_options std::experimental::filesystem::v1::operator~ (directory_options __x) noexcept`
- 
- `void std::experimental::filesystem::v1::copy (const path &__from, const path &__to, copy_options __options)`
  - `void std::experimental::filesystem::v1::copy (const path &__from, const path &__to, copy_options __options, error_code &) noexcept`
  - `bool std::experimental::filesystem::v1::copy_file (const path &__from, const path &__to, copy_options __option)`
  - `bool std::experimental::filesystem::v1::copy_file (const path &__from, const path &__to, copy_options __option, error_code &) noexcept`
  - `path std::experimental::filesystem::v1::current_path ()`
  - `file_status std::experimental::filesystem::v1::status (const path &)`
  - `file_status std::experimental::filesystem::v1::status (const path &, error_code &) noexcept`
  - `bool std::experimental::filesystem::v1::status_known (file_status) noexcept`
  - `file_status std::experimental::filesystem::v1::symlink_status (const path &)`
  - `file_status std::experimental::filesystem::v1::symlink_status (const path &, error_code &) noexcept`
  - `bool std::experimental::filesystem::v1::is_regular_file (file_status) noexcept`
  - `bool std::experimental::filesystem::v1::is_symlink (file_status) noexcept`

#### 6.251.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.252 fs\_path.h File Reference

#### 6.252.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 6.253 fs\_path.h File Reference

## Classes

- class [std::experimental::filesystem::v1::path](#)
- class [std::experimental::filesystem::v1::path::iterator](#)

## Namespaces

- [std](#)

## Functions

- void [std::experimental::filesystem::v1::swap](#) (path &\_\_lhs, path &\_\_rhs) noexcept
- size\_t [std::experimental::filesystem::v1::hash\\_value](#) (const path &\_\_p) noexcept
- bool [std::experimental::filesystem::v1::operator<](#) (const path &\_\_lhs, const path &\_\_rhs) noexcept
- bool [std::experimental::filesystem::v1::operator<=](#) (const path &\_\_lhs, const path &\_\_rhs) noexcept
- bool [std::experimental::filesystem::v1::operator>](#) (const path &\_\_lhs, const path &\_\_rhs) noexcept
- bool [std::experimental::filesystem::v1::operator>=](#) (const path &\_\_lhs, const path &\_\_rhs) noexcept
- bool [std::experimental::filesystem::v1::operator==](#) (const path &\_\_lhs, const path &\_\_rhs) noexcept
- bool [std::experimental::filesystem::v1::operator!=](#) (const path &\_\_lhs, const path &\_\_rhs) noexcept
- path [std::experimental::filesystem::v1::operator/](#) (const path &\_\_lhs, const path &\_\_rhs)
- template<typename \_CharT, typename \_Traits>  
basic\_ostream< \_CharT, \_Traits > & [std::experimental::filesystem::v1::operator<<](#) (basic\_ostream< \_CharT, \_Traits > &\_\_os, const path &\_\_p)
- template<typename \_CharT, typename \_Traits>  
basic\_istream< \_CharT, \_Traits > & [std::experimental::filesystem::v1::operator>>](#) (basic\_istream< \_CharT, \_Traits > &\_\_is, path &\_\_p)
- template<typename \_Source>  
path [std::experimental::filesystem::v1::u8path](#) (const \_Source &\_\_source)
- template<typename \_InputIterator>  
path [std::experimental::filesystem::v1::u8path](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)

## 6.253.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 6.253.2 Function Documentation

#### 6.253.2.1 hash\_value()

```
size_t std::experimental::filesystem::v1::hash_value (
    const path & __p ) [noexcept]
```

Compare paths.

#### 6.253.2.2 operator!=(())

```
bool std::experimental::filesystem::v1::operator!= (
    const path & __lhs,
    const path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 512 of file experimental/bits/fs\_path.h.

#### 6.253.2.3 operator/()

```
path std::experimental::filesystem::v1::operator/ (
    const path & __lhs,
    const path & __rhs ) [inline]
```

Append one path to another.

Definition at line 516 of file experimental/bits/fs\_path.h.

#### 6.253.2.4 operator<()

```
bool std::experimental::filesystem::v1::operator< (
    const path & __lhs,
    const path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 492 of file experimental/bits/fs\_path.h.

#### 6.253.2.5 operator<<()

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::experimental::filesystem::v1::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const path & __p )
```

Write a path to a stream.

Definition at line 526 of file experimental/bits/fs\_path.h.

#### 6.253.2.6 operator<=()

```
bool std::experimental::filesystem::v1::operator<= (
    const path & __lhs,
    const path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 496 of file experimental/bits/fs\_path.h.

#### 6.253.2.7 operator==()

```
bool std::experimental::filesystem::v1::operator== (
    const path & __lhs,
    const path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 508 of file experimental/bits/fs\_path.h.

#### 6.253.2.8 operator>()

```
bool std::experimental::filesystem::v1::operator> (
    const path & __lhs,
    const path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 500 of file experimental/bits/fs\_path.h.

#### 6.253.2.9 operator>=()

```
bool std::experimental::filesystem::v1::operator>= (
    const path & __lhs,
    const path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 504 of file experimental/bits/fs\_path.h.

**6.253.2.10 operator>>()**

```
template<typename _CharT , typename _Traits >
basic_istream<_CharT, _Traits>& std::experimental::filesystem::v1::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    path & __p )
```

Read a path from a stream.

Definition at line 538 of file experimental/bits/fs\_path.h.

**6.253.2.11 swap()**

```
void std::experimental::filesystem::v1::swap (
    path & __lhs,
    path & __rhs ) [inline], [noexcept]
```

Compare paths.

Definition at line 487 of file experimental/bits/fs\_path.h.

**6.253.2.12 u8path()** [1/2]

```
template<typename _Source >
path std::experimental::filesystem::v1::u8path (
    const _Source & __source ) [inline]
```

Compare paths.

Definition at line 551 of file experimental/bits/fs\_path.h.

References std::experimental::filesystem::v1::u8path().

Referenced by std::experimental::filesystem::v1::u8path().

**6.253.2.13 u8path()** [2/2]

```
template<typename _InputIterator >
path std::experimental::filesystem::v1::u8path (
    _InputIterator __first,
    _InputIterator __last ) [inline]
```

Compare paths.

Definition at line 563 of file experimental/bits/fs\_path.h.

References std::experimental::filesystem::v1::u8path().

## 6.254 **fstream File Reference**

### Classes

- class [std::basic\\_filebuf<\\_CharT, \\_Traits>](#)
- class [std::basic\\_fstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ifstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ofstream<\\_CharT, \\_Traits>](#)

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_FSTREAM`

### Functions

- `template<class _CharT, class _Traits>  
void std::swap (basic_filebuf<_CharT, _Traits> &__x, basic_filebuf<_CharT, _Traits> &__y)`
- `template<class _CharT, class _Traits>  
void std::swap (basic_ifstream<_CharT, _Traits> &__x, basic_ifstream<_CharT, _Traits> &__y)`
- `template<class _CharT, class _Traits>  
void std::swap (basic_ofstream<_CharT, _Traits> &__x, basic_ofstream<_CharT, _Traits> &__y)`
- `template<class _CharT, class _Traits>  
void std::swap (basic_fstream<_CharT, _Traits> &__x, basic_fstream<_CharT, _Traits> &__y)`

#### 6.254.1 Detailed Description

This is a Standard C++ Library header.

## 6.255 **fstream.tcc File Reference**

### Namespaces

- [std](#)

### Macros

- `#define _FSTREAM_TCC`



### 6.255.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<fstream>`.

## 6.256 funtexcept.h File Reference

### Namespaces

- [std](#)

### Functions

- void **std::\_\_throw\_bad\_alloc** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_cast** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_exception** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_function\_call** () \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_typeid** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_domain\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_future\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_invalid\_argument** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_ios\_failure** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_length\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_logic\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_out\_of\_range** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_out\_of\_range\_fmt** (const char \*,...) \_\_attribute\_\_((\_\_noreturn\_\_)) \_\_attribute\_\_((\_\_format\_\_(\_\_gnu\_printf\_\_
- void **std::\_\_throw\_overflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_range\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_runtime\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_system\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_underflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))

### 6.256.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

## 6.257 functional File Reference

## Classes

- struct [std::\\_Bind< \\_Signature >](#)
- struct [std::\\_Bind\\_result< \\_Result, \\_Signature >](#)
- class [std::\\_Mu< \\_Arg, \\_IsBindExp, \\_IsPlaceholder >](#)
- class [std::\\_Mu< \\_Arg, false, false >](#)
- class [std::\\_Mu< \\_Arg, false, true >](#)
- class [std::\\_Mu< \\_Arg, true, false >](#)
- class [std::\\_Mu< reference\\_wrapper< \\_Tp >, false, false >](#)
- class [std::\\_Not\\_fn< \\_Fn >](#)
- struct [std::\\_Placeholder< \\_Num >](#)
- struct [std::is\\_bind\\_expression< \\_Tp >](#)
- struct [std::is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< const \\_Bind< \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< const \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< const volatile \\_Bind< \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< const volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< volatile \\_Bind< \\_Signature > >](#)
- struct [std::is\\_bind\\_expression< volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [std::is\\_placeholder< \\_Tp >](#)
- struct [std::is\\_placeholder< \\_Placeholder< \\_Num > >](#)

## Namespaces

- [std](#)
- [std::placeholders](#)

## Macros

- `#define \_GLIBCXX\_DEPR\_BIND`
- `#define \_GLIBCXX\_FUNCTIONAL`
- `#define \_GLIBCXX\_NOT\_FN\_CALL\_OP\(\_QUALS\)`

## Typedefs

- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>  
using std::\_is\_socketlike = __or_< is_integral< _Tp2 >, is_enum< _Tp2 > >`
- `template<std::size_t __i, typename _Tuple >  
using std::\_Safe\_tuple\_element\_t = typename enable_if<(__i< tuple_size< _Tuple >::value), tuple_element<  
__i, _Tuple > >::type::type`

## Functions

- `template<std::size_t _Ind, typename... _Tp>`  
`auto std::__volget (volatile tuple< _Tp... > &__tuple) -> __tuple_element_t< _Ind, tuple< _Tp... >> volatile &`
- `template<std::size_t _Ind, typename... _Tp>`  
`auto std::__volget (const volatile tuple< _Tp... > &__tuple) -> __tuple_element_t< _Ind, tuple< _Tp... >>`  
`const volatile &`
- `template<typename _Func, typename... _BoundArgs>`  
`_Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _↵`  
`BoundArgs &&... __args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`_Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::* __pm) noexcept`

## Variables

- `const _Placeholder< 1 > std::placeholders::_1`
- `const _Placeholder< 10 > std::placeholders::_10`
- `const _Placeholder< 11 > std::placeholders::_11`
- `const _Placeholder< 12 > std::placeholders::_12`
- `const _Placeholder< 13 > std::placeholders::_13`
- `const _Placeholder< 14 > std::placeholders::_14`
- `const _Placeholder< 15 > std::placeholders::_15`
- `const _Placeholder< 16 > std::placeholders::_16`
- `const _Placeholder< 17 > std::placeholders::_17`
- `const _Placeholder< 18 > std::placeholders::_18`
- `const _Placeholder< 19 > std::placeholders::_19`
- `const _Placeholder< 2 > std::placeholders::_2`
- `const _Placeholder< 20 > std::placeholders::_20`
- `const _Placeholder< 21 > std::placeholders::_21`
- `const _Placeholder< 22 > std::placeholders::_22`
- `const _Placeholder< 23 > std::placeholders::_23`
- `const _Placeholder< 24 > std::placeholders::_24`
- `const _Placeholder< 25 > std::placeholders::_25`
- `const _Placeholder< 26 > std::placeholders::_26`
- `const _Placeholder< 27 > std::placeholders::_27`
- `const _Placeholder< 28 > std::placeholders::_28`
- `const _Placeholder< 29 > std::placeholders::_29`
- `const _Placeholder< 3 > std::placeholders::_3`
- `const _Placeholder< 4 > std::placeholders::_4`
- `const _Placeholder< 5 > std::placeholders::_5`
- `const _Placeholder< 6 > std::placeholders::_6`
- `const _Placeholder< 7 > std::placeholders::_7`
- `const _Placeholder< 8 > std::placeholders::_8`
- `const _Placeholder< 9 > std::placeholders::_9`

### 6.257.1 Detailed Description

This is a Standard C++ Library header.

## 6.258 functional File Reference

## Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose<\\_Operation1, \\_Operation2, \\_Operation3 >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun<\\_Result, \\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun<\\_Result, \\_Argument >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun<\\_Result >](#)
- struct [\\_\\_gnu\\_cxx::project1st<\\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::project2nd<\\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::select1st<\\_Pair >](#)
- struct [\\_\\_gnu\\_cxx::select2nd<\\_Pair >](#)
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose<\\_Operation1, \\_Operation2 >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define \_EXT\_FUNCTIONAL`

## Functions

- `template<class _Operation1, class _Operation2 >`  
`unary_compose<_Operation1, _Operation2 > \_\_gnu\_cxx::compose1 (const _Operation1 &__fn1, const _↔`  
`Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`  
`binary_compose<_Operation1, _Operation2, _Operation3 > \_\_gnu\_cxx::compose2 (const _Operation1 &__fn1,`  
`const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`  
`constant_void_fun<_Result > \_\_gnu\_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun<_Result, _Result > \_\_gnu\_cxx::constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant_binary_fun<_Result, _Result, _Result > \_\_gnu\_cxx::constant2 (const _Result &__val)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::plus<_Tp >)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::multiplies<_Tp >)`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem_fun1_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1 (_Ret(_Tp::*__f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`const_mem_fun1_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1 (_Ret(_Tp::*__f)(_Arg) const)`
- `template<class _Ret, class _Tp, class _Arg >`  
`mem_fun1_ref_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1\_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`const_mem_fun1_ref_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1\_ref (_Ret(_Tp::*__f)(_Arg) const)`

## 6.258.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.259 functional File Reference

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_experimental_boyer_moore_searching`
- `#define __cpp_lib_experimental_not_fn`
- `#define _GLIBCXX_EXPERIMENTAL_FUNCTIONAL`

### Typedefs

- `template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>`  
`using std::experimental::fundamentals\_v1::boyer\_moore\_base\_t = std::conditional\_t< std::is\_byte, like< _Val, _Pred >::value, \_\_boyer\_moore\_array\_base< _Diff, 256, _Pred >, \_\_boyer\_moore\_map\_base< _Val, _Diff, _Hash, _Pred > >`

### Functions

- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator\_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal\_to<>>`  
`boyer\_moore\_horspool\_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::fundamentals\_v1::make\_boyer\_moore`  
`(_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator\_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal\_to<>>`  
`boyer\_moore\_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::fundamentals\_v1::make\_boyer\_moore\_searcher`  
`(_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _ForwardIterator, typename _BinaryPredicate = std::equal\_to<>>`  
`default\_searcher<_ForwardIterator, _BinaryPredicate> std::experimental::fundamentals\_v1::make\_default\_searcher`  
`(_ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _Fn>`  
`auto std::experimental::fundamentals\_v2::not\_fn (_Fn &&__fn) noexcept(std::is\_nothrow\_constructible<std::decay\_t<_Fn>, _Fn && >::value)`

### Variables

- `template<typename _Tp>`  
`constexpr bool std::experimental::fundamentals\_v1::is\_bind\_expression\_v`
- `template<typename _Tp>`  
`constexpr int std::experimental::fundamentals\_v1::is\_placeholder\_v`

### 6.259.1 Detailed Description

This is a TS C++ Library header.

### 6.259.2 Function Documentation

#### 6.259.2.1 make\_boyer\_moore\_horspool\_searcher()

```
template<typename _RAIter , typename _Hash = std::hash<typename std::iterator_traits<_RAIter>↵
::value_type>, typename _BinaryPredicate = equal_to<>>
boyer_moore_horspool_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::fundamentals↵
_v1::make_boyer_moore_horspool_searcher (
    _RAIter __pat_first,
    _RAIter __pat_last,
    _Hash __hf = _Hash(),
    _BinaryPredicate __pred = _BinaryPredicate() ) [inline]
```

Generator function for boyer\_moore\_horspool\_searcher.

Definition at line 302 of file experimental/functional.

#### 6.259.2.2 make\_boyer\_moore\_searcher()

```
template<typename _RAIter , typename _Hash = std::hash<typename std::iterator_traits<_RAIter>↵
::value_type>, typename _BinaryPredicate = equal_to<>>
boyer_moore_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::fundamentals_v1::make↵
_boyer_moore_searcher (
    _RAIter __pat_first,
    _RAIter __pat_last,
    _Hash __hf = _Hash(),
    _BinaryPredicate __pred = _BinaryPredicate() ) [inline]
```

Generator function for boyer\_moore\_searcher.

Definition at line 292 of file experimental/functional.

#### 6.259.2.3 make\_default\_searcher()

```
template<typename _ForwardIterator , typename _BinaryPredicate = std::equal_to<>>
default_searcher<_ForwardIterator, _BinaryPredicate> std::experimental::fundamentals_v1::make_↵
default_searcher (
    _ForwardIterator __pat_first,
    _ForwardIterator __pat_last,
    _BinaryPredicate __pred = _BinaryPredicate() ) [inline]
```

Generator function for default\_searcher.

Definition at line 282 of file experimental/functional.

#### 6.259.2.4 not\_fn()

```
template<typename _Fn >
auto std::experimental::fundamentals_v2::not_fn (
    _Fn && __fn ) [inline], [noexcept]
```

[func.not\_fn] Function template not\_fn

Definition at line 371 of file experimental/functional.

### 6.259.3 Variable Documentation

#### 6.259.3.1 is\_bind\_expression\_v

```
template<typename _Tp >
constexpr bool std::experimental::fundamentals_v1::is_bind_expression_v
```

Variable template for std::is\_bind\_expression.

Definition at line 60 of file experimental/functional.

#### 6.259.3.2 is\_placeholder\_v

```
template<typename _Tp >
constexpr int std::experimental::fundamentals_v1::is_placeholder_v
```

Variable template for std::is\_placeholder.

Definition at line 64 of file experimental/functional.

## 6.260 functional\_hash.h File Reference

### Classes

- struct [std::hash< \\_Tp >](#)
- struct [std::hash< \\_Tp >](#)
- struct [std::hash< \\_Tp \\* >](#)
- struct [std::hash< bool >](#)
- struct [std::hash< char >](#)
- struct [std::hash< char16\\_t >](#)
- struct [std::hash< char32\\_t >](#)
- struct [std::hash< double >](#)
- struct [std::hash< float >](#)
- struct [std::hash< int >](#)
- struct [std::hash< long >](#)
- struct [std::hash< long double >](#)
- struct [std::hash< long long >](#)
- struct [std::hash< short >](#)
- struct [std::hash< signed char >](#)
- struct [std::hash< unsigned char >](#)
- struct [std::hash< unsigned int >](#)
- struct [std::hash< unsigned long >](#)
- struct [std::hash< unsigned long long >](#)
- struct [std::hash< unsigned short >](#)
- struct [std::hash< wchar\\_t >](#)

## Namespaces

- [std](#)

## Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

## 6.260.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.261 functions.h File Reference

## Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator](#)< [\\_Iterator](#), [\\_Sequence](#) >

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Functions

- `template<typename \_Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const \_Iterator &)`
- `template<typename \_Tp >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const \_Tp * __ptr)`
- `template<typename \_ForwardIterator , typename \_Tp >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (\_ForwardIterator __first, \_ForwardIterator __last, const \_Tp & __value)`
- `template<typename \_ForwardIterator , typename \_Tp , typename \_Pred >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (\_ForwardIterator __first, \_ForwardIterator __last, const \_Tp & __value, \_Pred __pred)`
- `template<typename \_ForwardIterator , typename \_Tp >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (\_ForwardIterator __first, \_ForwardIterator __last, const \_Tp & __value)`
- `template<typename \_ForwardIterator , typename \_Tp , typename \_Pred >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (\_ForwardIterator __first, \_ForwardIterator __last, const \_Tp & __value, \_Pred __pred)`
- `template<typename \_Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const \_Iterator &)`
- `template<typename \_Tp >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const \_Tp * __ptr)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const void *)`



- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::true\_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::true\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false\_type)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * __gnu_debug::__check_string (const _CharT * __s, const _Integer & __n __attribute__((\_\_unused)))`
- `template<typename _CharT >`  
`const _CharT * __gnu_debug::__check_string (const _CharT * __s)`
- `template<typename _InputIterator >`  
`_InputIterator __gnu_debug::__check_valid_range (const _InputIterator &__first, const _InputIterator &__last __attribute__((\_\_unused)))`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator (const _Safe_iterator< _Iterator, _Sequence > &__it, _InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Integral >`  
`bool __gnu_debug::__foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence > &, _Integral, std::true\_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence > &__it, _InputIterator __other, _InputIterator __other_end, std::false\_type)`
- `template<typename _Iterator, typename _Sequence, typename _OtherIterator >`  
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _Safe_iterator< _OtherIterator, _Sequence > &__other, const _Safe_iterator< _OtherIterator, _Sequence > &)`

- `template<typename _Iterator, typename _Sequence, typename _OtherIterator, typename _OtherSequence >`  
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _Safe_↵`  
`_iterator< _OtherIterator, _OtherSequence > &, const _Safe_iterator< _OtherIterator, _OtherSequence > &)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _↵`  
`InputIterator &__other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _↵`  
`InputIterator &__other, const _InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence > &, const _Input_↵`  
`Iterator &, const _InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::__foreign_iterator_aux4 (const _Safe_iterator< _Iterator, _Sequence > &__it, const type-↵`  
`name _Sequence::value_type *__other)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::__foreign_iterator_aux4 (const _Safe_iterator< _Iterator, _Sequence > &,...)`
- `template<typename _Iterator >`  
`bool __gnu_debug::__is_irreflexive (_Iterator __it)`
- `template<typename _Iterator, typename _Pred >`  
`bool __gnu_debug::__is_irreflexive_pred (_Iterator __it, _Pred __pred)`

#### 6.261.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.262 future File Reference

### Classes

- `class std::__basic_future< _Res >`
- `struct std::__future_base`
- `struct std::__future_base::Result< _Res >`
- `struct std::__future_base::Result< _Res & >`
- `struct std::__future_base::Result< void >`
- `struct std::__future_base::Result_alloc< _Res, _Alloc >`
- `struct std::__future_base::Result_base`
- `class std::future< _Res >`
- `class std::future< _Res >`
- `class std::future< _Res & >`
- `class std::future< void >`
- `class std::future_error`
- `struct std::is_error_code_enum< future_errc >`
- `class std::packaged_task< _Res(_ArgTypes...)>`
- `class std::promise< _Res >`
- `class std::promise< _Res >`
- `class std::promise< _Res & >`
- `class std::promise< void >`
- `class std::shared_future< _Res >`
- `class std::shared_future< _Res >`
- `class std::shared_future< _Res & >`
- `class std::shared_future< void >`

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_FUTURE`

## Typedefs

- `template<typename _Fn, typename... _Args>  
using std::\_\_async\_result\_of = typename result_of< typename decay< _Fn >::type(typename decay< _Args  
>::type...)>::type`

## Enumerations

- `enum std::future\_errc { future\_already\_retrieved, promise\_already\_satisfied, no\_state, broken\_promise }`
- `enum std::future\_status { ready, timeout, deferred }`
- `enum std::launch { async, deferred }`

## Functions

- `template<typename _Signature, typename _Fn, typename _Alloc >  
static shared_ptr< __future_base::__Task_state_base< _Signature > > std::\_\_create\_task\_state (_Fn &&__fn,  
const _Alloc &__a)`
- `template<typename _Fn, typename... _Args>  
future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- `template<typename _Fn, typename... _Args>  
future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)`
- `const error_category & std::future\_category () noexcept`
- `error_code std::make\_error\_code (future_errc __errc) noexcept`
- `error_condition std::make\_error\_condition (future_errc __errc) noexcept`
- `constexpr launch std::operator& (launch __x, launch __y)`
- `launch & std::operator&= (launch &__x, launch __y)`
- `constexpr launch std::operator^ (launch __x, launch __y)`
- `launch & std::operator^= (launch &__x, launch __y)`
- `constexpr launch std::operator| (launch __x, launch __y)`
- `launch & std::operator|= (launch &__x, launch __y)`
- `constexpr launch std::operator~ (launch __x)`
- `template<typename _Res >  
void std::swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`
- `template<typename _Res, typename... _ArgTypes>  
void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noex-  
cept`

### 6.262.1 Detailed Description

This is a Standard C++ Library header.

## 6.263 gp\_ht\_map.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_GP_HASH_NAME`
- `#define PB_DS_GP_HASH_TRAITS_BASE`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_PROBE_FN_C_DEC`

### Variables

- `empty_entry_status`
- `erased_entry_status`
- `valid_entry_status`

#### 6.263.1 Detailed Description

Contains an implementation class for general probing hash.

## 6.264 gsllice.h File Reference

### Classes

- class [std::gsllice](#)

### Namespaces

- [std](#)

#### 6.264.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.265 `gslice_array.h` File Reference

### Classes

- class `std::gslice_array<_Tp>`

### Namespaces

- `std`

### Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

#### 6.265.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.266 `hash_bytes.h` File Reference

### Namespaces

- `std`

### Functions

- `size_t std::_Fnv_hash_bytes` (const void \* \_\_ptr, size\_t \_\_len, size\_t \_\_seed)
- `size_t std::_Hash_bytes` (const void \* \_\_ptr, size\_t \_\_len, size\_t \_\_seed)

#### 6.266.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.267 `hash_eq_fn.hpp` File Reference

### Classes

- struct `__gnu_pbds::detail::hash_eq_fn<Key, Eq_Fn, _Alloc, Store_Hash>`
- struct `__gnu_pbds::detail::hash_eq_fn<Key, Eq_Fn, _Alloc, false>`
- struct `__gnu_pbds::detail::hash_eq_fn<Key, Eq_Fn, _Alloc, true>`

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 6.267.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

## 6.268 hash\_exponential\_size\_policy\_imp.hpp File Reference

## 6.268.1 Detailed Description

Contains a resize size policy implementation.

## 6.269 hash\_fun.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `size_t __gnu_cxx::__stl_hash_string (const char *__s)`

## 6.269.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.270 hash\_load\_check\_resize\_trigger\_imp.hpp File Reference

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_ASSERT_VALID(X)`

## 6.270.1 Detailed Description

Contains a resize trigger implementation.

## 6.271 hash\_load\_check\_resize\_trigger\_size\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, Hold\\_Size >](#)
- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.271.1 Detailed Description

Contains an base holding size for some resize policies.

## 6.272 hash\_map File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_map< \\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap< \\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define _BACKWARD_HASH_MAP`

### Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
void __gnu_cxx::swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
void __gnu_cxx::swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`

## 6.272.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.273 hash\_policy.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#)
- class [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::quadratic\\_probe\\_fn< Size\\_Type >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`



#### Enumerations

- enum { **num\_distinct\_sizes\_32\_bit**, **num\_distinct\_sizes\_64\_bit**, **num\_distinct\_sizes** }

#### Variables

- static const std::size\_t **\_\_gnu\_pbds::detail::g\_a\_sizes** [num\_distinct\_sizes\_64\_bit]

#### 6.273.1 Detailed Description

Contains hash-related policies.

### 6.274 hash\_prime\_size\_policy\_imp.hpp File Reference

#### Enumerations

- enum { **num\_distinct\_sizes\_32\_bit**, **num\_distinct\_sizes\_64\_bit**, **num\_distinct\_sizes** }

#### Variables

- static const std::size\_t **detail::g\_a\_sizes** [num\_distinct\_sizes\_64\_bit]

#### 6.274.1 Detailed Description

Contains a resize size policy implementation.

### 6.275 hash\_set File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::hash\\_multiset<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_set<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

#### Macros

- **#define \_BACKWARD\_HASH\_SET**

## Functions

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

## 6.275.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.276 hash\_standard\_resize\_policy\_imp.hpp File Reference

## 6.276.1 Detailed Description

Contains a resize policy implementation.

## 6.277 hashtable.h File Reference

## Classes

- [class std::\\_Hashtable< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits >](#)

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Tp, typename _Hash >`  
`using std::__cache_default = __not_< __and_< __is_fast_hash< _Hash >, __is_nothrow_invocable< const _Hash &, const _Tp > >>`

### 6.277.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 6.278 hashtable.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Enumerations

- `enum { _S_num_primes }`

### Functions

- `unsigned long __gnu_cxx::__stl_next_prime (unsigned long __n)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >  
bool __gnu_cxx::operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >  
bool __gnu_cxx::operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >  
void __gnu_cxx::swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`

### 6.278.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.279 hashtable\_policy.h File Reference

### Classes

- `struct std::__detail::__Default_ranged_hash`
- `struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >`
- `struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >`
- `struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >`
- `struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_key >`
- `struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- `struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- `struct std::__detail::__Equality_base`

- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, false >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, true >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >
- struct std::\_\_detail::Hash\_node< \_Value, \_Cache\_hash\_code >
- struct std::\_\_detail::Hash\_node< \_Value, false >
- struct std::\_\_detail::Hash\_node< \_Value, true >
- struct std::\_\_detail::Hash\_node\_base
- struct std::\_\_detail::Hash\_node\_value\_base< \_Value >
- struct std::\_\_detail::Hashtable\_alloc< \_NodeAlloc >
- struct std::\_\_detail::Hashtable\_alloc< \_NodeAlloc >
- struct std::\_\_detail::Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, \_\_use\_ebo >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, false >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, true >
- struct std::\_\_detail::Hashtable\_traits< \_Cache\_hash\_code, \_Constant\_iterators, \_Unique\_keys >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Constant\_iterators >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- struct std::\_\_detail::Local\_const\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Local\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct std::\_\_detail::Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, true >
- struct std::\_\_detail::Map\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >
- struct std::\_\_detail::Map\_base< \_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::Map\_base< \_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::Mask\_range\_hashing
- struct std::\_\_detail::Mod\_range\_hashing
- struct std::\_\_detail::Node\_const\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >
- struct std::\_\_detail::Node\_iterator\_base< \_Value, \_Cache\_hash\_code >
- struct std::\_\_detail::Power2\_rehash\_policy
- struct std::\_\_detail::Prime\_rehash\_policy
- struct std::\_\_detail::Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, typename \_Key >
- struct std::\_\_detail::Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, std::false\_type >
- struct std::\_\_detail::Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, std::true\_type >
- class std::Hashtable< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Typedefs

- `template<typename _Policy >`  
`using std::__detail::__has_load_factor = typename _Policy::__has_load_factor`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash >`  
`using std::__detail::__hash_code_for_local_iter = Hash_code_storage< _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > >`

## Functions

- `_GLIBCXX14_CONSTEXPR std::size_t std::\_\_detail::\_\_clp2 (std::size_t __n) noexcept`
- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::input\_iterator\_tag)`
- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::forward\_iterator\_tag)`
- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _Value , bool _Cache_hash_code>  
bool std::__detail::operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const ↵  
_Node_iterator_base< _Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>  
bool std::__detail::operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, ↵  
__cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _Value , bool _Cache_hash_code>  
bool std::__detail::operator== (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const ↵  
_Node_iterator_base< _Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>  
bool std::__detail::operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, ↵  
__cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`

### 6.279.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 6.280 `helper_functions.h` File Reference

### Namespaces

- [\\_\\_gnu\\_debug](#)

### Enumerations

- enum [\\_\\_gnu\\_debug::Distance\\_precision](#) { `__dp_none`, `__dp_equality`, `__dp_sign`, `__dp_exact` }

## Functions

- `template<typename _Iterator >`  
`_Iterator gnu_debug::__base (_Iterator __it)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type gnu_debug::__get_distance (const _Iterator &__lhs, const _Iterator &__rhs, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type gnu_debug::__get_distance (const _Iterator &__lhs, const _Iterator &__rhs, std::input\_iterator\_tag)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type gnu_debug::__get_distance (const _Iterator &__lhs, const _Iterator &__rhs)`
- `template<typename _Iterator >`  
`_Iterator gnu_debug::__unsafe (_Iterator __it)`
- `template<typename _InputIterator >`  
`bool gnu_debug::__valid_range (const _InputIterator &__first, const _InputIterator &__last, typename _Distance_traits< _InputIterator >::__type &__dist)`
- `template<typename _InputIterator >`  
`bool gnu_debug::__valid_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Integral >`  
`bool gnu_debug::__valid_range_aux (const _Integral &, const _Integral &, typename _Distance_traits< _Integral >::__type &__dist, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool gnu_debug::__valid_range_aux (const _InputIterator &__first, const _InputIterator &__last, typename _Distance_traits< _InputIterator >::__type &__dist, std::\_\_false\_type)`

## 6.280.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.281 indirect\_array.h File Reference

## Classes

- class [std::indirect\\_array< \\_Tp >](#)

## Namespaces

- [std](#)

## Macros

- `#define DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

#### 6.281.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 6.282 info\_fn\_imps.hpp File Reference

#### 6.282.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.283 info\_fn\_imps.hpp File Reference

#### 6.283.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 6.284 info\_fn\_imps.hpp File Reference

#### 6.284.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container info related functions.

### 6.285 info\_fn\_imps.hpp File Reference

#### 6.285.1 Detailed Description

Contains implementations of `gp_ht_map_`'s entire container info related functions.

### 6.286 info\_fn\_imps.hpp File Reference

#### 6.286.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

### 6.287 info\_fn\_imps.hpp File Reference

#### 6.287.1 Detailed Description

Contains implementations of `lu_map_`.

## 6.288 info\_fn\_imps.hpp File Reference

### 6.288.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.289 info\_fn\_imps.hpp File Reference

### 6.289.1 Detailed Description

Contains an implementation class for pat\_trie.

## 6.290 info\_fn\_imps.hpp File Reference

### 6.290.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.291 info\_fn\_imps.hpp File Reference

### 6.291.1 Detailed Description

Contains an implementation.

## 6.292 initializer\_list File Reference

### Classes

- class [std::initializer\\_list<\\_E>](#)

### Namespaces

- [std](#)

### Functions

- [template<class \\_Tp>](#)  
constexpr const \_Tp \* [std::begin](#) (initializer\_list<\_Tp> \_\_ils) noexcept
- [template<class \\_Tp>](#)  
constexpr const \_Tp \* [std::end](#) (initializer\_list<\_Tp> \_\_ils) noexcept



#### 6.292.1 Detailed Description

This is a Standard C++ Library header.

### 6.293 `insert_fn_imps.hpp` File Reference

#### 6.293.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 6.294 `insert_fn_imps.hpp` File Reference

#### 6.294.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

### 6.295 `insert_fn_imps.hpp` File Reference

#### 6.295.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 6.296 `insert_fn_imps.hpp` File Reference

#### 6.296.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions.

### 6.297 `insert_fn_imps.hpp` File Reference

#### 6.297.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions.

### 6.298 `insert_fn_imps.hpp` File Reference

#### 6.298.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.299 insert\_fn\_imps.hpp File Reference

### 6.299.1 Detailed Description

Contains implementations of lu\_map\_.

## 6.300 insert\_fn\_imps.hpp File Reference

### 6.300.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 6.301 insert\_fn\_imps.hpp File Reference

### 6.301.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.302 insert\_fn\_imps.hpp File Reference

### 6.302.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.303 insert\_fn\_imps.hpp File Reference

### 6.303.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 6.304 insert\_fn\_imps.hpp File Reference

### 6.304.1 Detailed Description

Contains an implementation class for splay\_tree\_.

### 6.305 `insert_fn_imps.hpp` File Reference

#### 6.305.1 Detailed Description

Contains an implementation for `thin_heap_`.

### 6.306 `insert_join_fn_imps.hpp` File Reference

#### 6.306.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 6.307 `insert_no_store_hash_fn_imps.hpp` File Reference

#### 6.307.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is not stored.

### 6.308 `insert_no_store_hash_fn_imps.hpp` File Reference

#### 6.308.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions, when the hash value is not stored.

### 6.309 `insert_store_hash_fn_imps.hpp` File Reference

#### 6.309.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is stored.

### 6.310 `insert_store_hash_fn_imps.hpp` File Reference

#### 6.310.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions, when the hash value is stored.

### 6.311 `invoke.h` File Reference

#### Namespaces

- [std](#)

## Functions

- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>  
constexpr _Up && std::__invfwd (typename remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>  
constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (_Callable &&__fn, _Args &&... __args)  
noexcept(__is_nothrow_invocable<_Callable, _Args...>::value)`
- `template<typename _Res, typename _Fn, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args) ←`
- `template<typename _Res, typename _MemPtr, typename _Tp>  
constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>  
constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`

### 6.311.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.312 iomanip File Reference

### Namespaces

- `std`

### Macros

- `#define __cpp_lib_quoted_string_io`
- `#define _GLIBCXX_IOMANIP`

## Functions

- `template<typename _MoneyT>  
_Get_money<_MoneyT> std::get_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT>  
_Get_time<_CharT> std::get_time (std::tm *__tmb, const _CharT *__fmt)`
- `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__os, _↵  
Resetiosflags __f)`
- `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__os, _↵  
__f)`

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _`  
`Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_`  
`money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_time<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _`  
`CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money<`  
`_MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_time<`  
`_CharT > __f)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT >`  
`_Put_time< _CharT > std::put_time (const std::tm * __tmb, const _CharT * __fmt)`
- `template<typename _CharT >`  
`auto std::quoted (const _CharT * __string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto std::quoted (const basic_string< _CharT, _Traits, _Alloc > & __string, _CharT __delim=_CharT(""), _CharT`  
`__escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto std::quoted (basic_string< _CharT, _Traits, _Alloc > & __string, _CharT __delim=_CharT(""), _CharT`  
`__escape = _CharT("\\"))`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`

- `template<typename _CharT > _Setfill< _CharT > std::setfill ( _CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

#### 6.312.1 Detailed Description

This is a Standard C++ Library header.

### 6.313 ios File Reference

#### Macros

- `#define \_GLIBCXX\_IOS`

#### 6.313.1 Detailed Description

This is a Standard C++ Library header.

### 6.314 ios\_base.h File Reference

#### Classes

- class [std::ios\\_base](#)
- class [std::ios\\_base::failure](#)

#### Namespaces

- [std](#)

#### Enumerations

- enum `_ios_Fmtflags` {  
`_S_boolalpha, _S_dec, _S_fixed, _S_hex,`  
`_S_internal, _S_left, _S_oct, _S_right,`  
`_S_scientific, _S_showbase, _S_showpoint, _S_showpos,`  
`_S_skipws, _S_unitbuf, _S_uppercase, _S_adjustfield,`  
`_S_basefield, _S_floatfield, _S_ios_fmtflags_end, _S_ios_fmtflags_max,`  
`_S_ios_fmtflags_min }`
- enum `_ios_Iostate` {  
`_S_goodbit, _S_badbit, _S_eofbit, _S_failbit,`  
`_S_ios_iostate_end, _S_ios_iostate_max, _S_ios_iostate_min }`
- enum `_ios_Openmode` {  
`_S_app, _S_ate, _S_bin, _S_in,`  
`_S_out, _S_trunc, _S_ios_openmode_end, _S_ios_openmode_max,`  
`_S_ios_openmode_min }`
- enum `_ios_Seekdir` { `_S_beg, _S_cur, _S_end, _S_ios_seekdir_end }`
- enum `std::io\_errc` { `stream` }

## Functions

- `ios_base & std::boolalpha (ios_base & __base)`
- `ios_base & std::dec (ios_base & __base)`
- `ios_base & std::defaultfloat (ios_base & __base)`
- `ios_base & std::fixed (ios_base & __base)`
- `ios_base & std::hex (ios_base & __base)`
- `ios_base & std::hexfloat (ios_base & __base)`
- `ios_base & std::internal (ios_base & __base)`
- `const error_category & std::iostream_category () noexcept`
- `ios_base & std::left (ios_base & __base)`
- `error_code std::make_error_code (io_errc __e) noexcept`
- `error_condition std::make_error_condition (io_errc __e) noexcept`
- `ios_base & std::noboolalpha (ios_base & __base)`
- `ios_base & std::noshowbase (ios_base & __base)`
- `ios_base & std::noshowpoint (ios_base & __base)`
- `ios_base & std::noshowpos (ios_base & __base)`
- `ios_base & std::noskipws (ios_base & __base)`
- `ios_base & std::nounitbuf (ios_base & __base)`
- `ios_base & std::nouppercase (ios_base & __base)`
- `ios_base & std::oct (ios_base & __base)`
- `constexpr _ios_Fmtflags std::operator& (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator& (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_istate std::operator& (_ios_istate __a, _ios_istate __b)`
- `const _ios_Fmtflags & std::operator&= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator&= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_istate & std::operator&= (_ios_istate & __a, _ios_istate __b)`
- `constexpr _ios_Fmtflags std::operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator^ (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_istate std::operator^ (_ios_istate __a, _ios_istate __b)`
- `const _ios_Fmtflags & std::operator^= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator^= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_istate & std::operator^= (_ios_istate & __a, _ios_istate __b)`
- `constexpr _ios_Fmtflags std::operator| (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator| (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_istate std::operator| (_ios_istate __a, _ios_istate __b)`
- `const _ios_Fmtflags & std::operator|= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator|= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_istate & std::operator|= (_ios_istate & __a, _ios_istate __b)`
- `constexpr _ios_Fmtflags std::operator~ (_ios_Fmtflags __a)`
- `constexpr _ios_Openmode std::operator~ (_ios_Openmode __a)`
- `constexpr _ios_istate std::operator~ (_ios_istate __a)`
- `ios_base & std::right (ios_base & __base)`
- `ios_base & std::scientific (ios_base & __base)`
- `ios_base & std::showbase (ios_base & __base)`
- `ios_base & std::showpoint (ios_base & __base)`
- `ios_base & std::showpos (ios_base & __base)`
- `ios_base & std::skipws (ios_base & __base)`
- `ios_base & std::unitbuf (ios_base & __base)`
- `ios_base & std::uppercase (ios_base & __base)`

### 6.314.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 6.315 iosfwd File Reference

### Classes

- class [std::basic\\_filebuf< \\_CharT, \\_Traits >](#)
- class [std::basic\\_fstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ifstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ios< \\_CharT, \\_Traits >](#)
- class [std::basic\\_iostream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_istream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#)
- class [std::basic\\_ofstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ostream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ostreamstream< \\_CharT, \\_Traits, \\_Alloc >](#)
- class [std::basic\\_streambuf< \\_CharT, \\_Traits >](#)
- class [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#)
- class [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#)
- class [std::istreambuf\\_iterator< \\_CharT, \\_Traits >](#)
- class [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

### Macros

- `#define \_GLIBCXX\_IOSFWD`

### Typedefs

- typedef [basic\\_filebuf< char >](#) [std::filebuf](#)
- typedef [basic\\_fstream< char >](#) [std::fstream](#)
- typedef [basic\\_ifstream< char >](#) [std::ifstream](#)
- typedef [basic\\_ios< char >](#) [std::ios](#)
- typedef [basic\\_iostream< char >](#) [std::iostream](#)
- typedef [basic\\_istream< char >](#) [std::istream](#)
- typedef [basic\\_istreamstream< char >](#) [std::istreamstream](#)
- typedef [basic\\_ofstream< char >](#) [std::ofstream](#)
- typedef [basic\\_ostream< char >](#) [std::ostream](#)
- typedef [basic\\_ostreamstream< char >](#) [std::ostreamstream](#)
- typedef [basic\\_streambuf< char >](#) [std::streambuf](#)



- typedef basic\_stringbuf< char > [std::stringbuf](#)
- typedef basic\_stringstream< char > [std::stringstream](#)
- typedef basic\_filebuf< wchar\_t > [std::wfilebuf](#)
- typedef basic\_fstream< wchar\_t > [std::wfstream](#)
- typedef basic\_ifstream< wchar\_t > [std::wifstream](#)
- typedef basic\_ios< wchar\_t > [std::wios](#)
- typedef basic\_iostream< wchar\_t > [std::wiostream](#)
- typedef basic\_istream< wchar\_t > [std::wistream](#)
- typedef basic\_istreamstream< wchar\_t > [std::wistreamstream](#)
- typedef basic\_ofstream< wchar\_t > [std::wofstream](#)
- typedef basic\_ostream< wchar\_t > [std::wostream](#)
- typedef basic\_ostreamstream< wchar\_t > [std::wostreamstream](#)
- typedef basic\_streambuf< wchar\_t > [std::wstreambuf](#)
- typedef basic\_stringbuf< wchar\_t > [std::wstringbuf](#)
- typedef basic\_stringstream< wchar\_t > [std::wstringstream](#)

#### 6.315.1 Detailed Description

This is a Standard C++ Library header.

### 6.316 iostream File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define \_GLIBCXX\_IOSTREAM`

#### Variables

- static ios\_base::Init [std::\\_\\_ioinit](#)

#### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- istream [std::cin](#)
- ostream [std::cout](#)
- ostream [std::cerr](#)
- ostream [std::clog](#)
- wistream [std::wcin](#)
- wostream [std::wcout](#)
- wostream [std::wcerr](#)
- wostream [std::wclog](#)

## 6.316.1 Detailed Description

This is a Standard C++ Library header.

## 6.317 istream File Reference

## Classes

- class [std::basic\\_iostream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_istream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_istream< \\_CharT, \\_Traits >::sentry](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_ISTREAM`

## Typedefs

- `template<typename _Tp >`  
`using std::__do_is_convertible_to_basic_istream_impl = decltype(__is_convertible_to_basic_istream_↔`  
`test(declval< typename remove_reference< _Tp >::type * >()))`
- `template<typename _Istream >`  
`using std::__rvalue_istream_type = typename __is_convertible_to_basic_istream< _Istream >::__istream_↔`  
`type`

## Functions

- `template<typename _Ch, typename _Up >`  
`basic_istream< _Ch, _Up > & std::__is_convertible_to_basic_istream_test (basic_istream< _Ch, _Up > *)`
- `template<typename _Istream, typename _Tp >`  
`enable_if< __and< __not< is_lvalue_reference< _Istream > >, __is_convertible_to_basic_istream< _↔`  
`Istream >, __is_extractable< __rvalue_istream_type< _Istream >, _Tp && > >::value, __rvalue_istream_↔`  
`type< _Istream > >::type std::operator>> (_Istream && __is, _Tp && __x)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > & __is)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __in, _CharT & __c)`

- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`

#### 6.317.1 Detailed Description

This is a Standard C++ Library header.

### 6.318 `istream.tcc` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _ISTREAM_TCC`

#### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`

## 6.318.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<istream>`.

## 6.319 iterator File Reference

## Macros

- `#define _GLIBCXX_ITERATOR`

## 6.319.1 Detailed Description

This is a Standard C++ Library header.

## 6.320 iterator File Reference

## Namespaces

- [`\_\_gnu\_cxx`](#)

## Macros

- `#define _EXT_ITERATOR`

## Functions

- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

## 6.320.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.321 iterator File Reference

### Classes

- class [std::experimental::fundamentals\\_v2::ostream\\_joiner<\\_DelimT, \\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_experimental_ostream_joiner`
- `#define _GLIBCXX_EXPERIMENTAL_ITERATOR`

### Functions

- `template<typename _CharT, typename _Traits, typename _DelimT >`  
`ostream_joiner<decay_t<_DelimT>, _CharT, _Traits> std::experimental::fundamentals\_v2::make\_ostream\_joiner`  
`(basic_ostream<_CharT, _Traits> &__os, _DelimT &&__delimiter)`

#### 6.321.1 Detailed Description

This is a TS C++ Library header.

#### 6.321.2 Function Documentation

##### 6.321.2.1 `make_ostream_joiner()`

```
template<typename _CharT, typename _Traits, typename _DelimT >
ostream_joiner<decay_t<_DelimT>, _CharT, _Traits> std::experimental::fundamentals_v2::make_↵
ostream_joiner (
    basic_ostream<_CharT, _Traits> & __os,
    _DelimT && __delimiter ) [inline]
```

Object generator for `ostream_joiner`.

Definition at line 103 of file `experimental/iterator`.

## 6.322 iterator.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair<\\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple<\\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### 6.322.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

## 6.323 iterator.hpp File Reference

### Classes

- class [iterator\\_](#)

#### 6.323.1 Detailed Description

Contains an `iterator_` class used for ranging over the elements of the table.

## 6.324 iterator\_fn\_imps.hpp File Reference

#### 6.324.1 Detailed Description

Contains implementations of `gp_ht_map_`'s iterators related functions, e.g., `begin()`.

## 6.325 iterator\_tracker.h File Reference

### Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (typename __iterator_tracker< _Iterator,`  
`_Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__profile::operator- (const __iterator_↵`  
`tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type std::__profile::operator- (const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`  
`_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`  
`_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`  
`_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`  
`tracker< _Iterator, _Sequence > &__rhs) noexcept`

## 6.325.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.326 iterators\_fn\_imps.hpp File Reference

### 6.326.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 6.327 iterators\_fn\_imps.hpp File Reference

### 6.327.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 6.328 iterators\_fn\_imps.hpp File Reference

### 6.328.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s iterators related functions, e.g., begin().

## 6.329 iterators\_fn\_imps.hpp File Reference

### 6.329.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 6.330 iterators\_fn\_imps.hpp File Reference

### 6.330.1 Detailed Description

Contains implementations of lu\_map\_.

## 6.331 iterators\_fn\_imps.hpp File Reference

### 6.331.1 Detailed Description

Contains an implementation class for ov\_tree\_.



### 6.332 iterators\_fn\_imps.hpp File Reference

#### 6.332.1 Detailed Description

Contains an implementation class for pat\_trie.

### 6.333 left\_child\_next\_sibling\_heap.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.333.1 Detailed Description

Contains an implementation class for a basic heap.

### 6.334 lfts\_config.h File Reference

#### 6.334.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 6.335 limits File Reference

### Classes

- struct [std::\\_\\_numeric\\_limits\\_base](#)
- struct [std::numeric\\_limits< \\_Tp >](#)
- struct [std::numeric\\_limits< bool >](#)
- struct [std::numeric\\_limits< char >](#)
- struct [std::numeric\\_limits< char16\\_t >](#)
- struct [std::numeric\\_limits< char32\\_t >](#)
- struct [std::numeric\\_limits< double >](#)
- struct [std::numeric\\_limits< float >](#)
- struct [std::numeric\\_limits< int >](#)
- struct [std::numeric\\_limits< long >](#)
- struct [std::numeric\\_limits< long double >](#)
- struct [std::numeric\\_limits< long long >](#)
- struct [std::numeric\\_limits< short >](#)
- struct [std::numeric\\_limits< signed char >](#)
- struct [std::numeric\\_limits< unsigned char >](#)
- struct [std::numeric\\_limits< unsigned int >](#)
- struct [std::numeric\\_limits< unsigned long >](#)
- struct [std::numeric\\_limits< unsigned long long >](#)
- struct [std::numeric\\_limits< unsigned short >](#)
- struct [std::numeric\\_limits< wchar\\_t >](#)

### Namespaces

- [std](#)

### Macros

- `#define \_\_glibcxx\_digits\(T\)`
- `#define \_\_glibcxx\_digits10\(T\)`
- `#define \_\_glibcxx\_digits10\_b\(T, B\)`
- `#define \_\_glibcxx\_digits\_b\(T, B\)`
- `#define \_\_glibcxx\_double\_has\_denorm\_loss`
- `#define \_\_glibcxx\_double\_tinyness\_before`
- `#define \_\_glibcxx\_double\_traps`
- `#define \_\_glibcxx\_float\_has\_denorm\_loss`
- `#define \_\_glibcxx\_float\_tinyness\_before`
- `#define \_\_glibcxx\_float\_traps`
- `#define \_\_glibcxx\_integral\_traps`
- `#define \_\_glibcxx\_long\_double\_has\_denorm\_loss`
- `#define \_\_glibcxx\_long\_double\_tinyness\_before`
- `#define \_\_glibcxx\_long\_double\_traps`
- `#define \_\_glibcxx\_max\(T\)`
- `#define \_\_glibcxx\_max\_b\(T, B\)`
- `#define \_\_glibcxx\_max\_digits10\(T\)`
- `#define \_\_glibcxx\_min\(T\)`

- `#define __glibcxx_min_b(T, B)`
- `#define __glibcxx_signed(T)`
- `#define __glibcxx_signed_b(T, B)`
- `#define __INT_N(TYPE, BITSIZE, EXT, UEXT)`
- `#define __INT_N_201103(TYPE)`
- `#define __INT_N_U201103(TYPE)`
- `#define _GLIBCXX_NUMERIC_LIMITS`

#### Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate`, `std::denorm_absent`, `std::denorm_present` }
- enum `std::float_round_style` {  
    `round_indeterminate`, `std::round_toward_zero`, `std::round_to_nearest`, `std::round_toward_infinity`,  
    `std::round_toward_neg_infinity` }

#### 6.335.1 Detailed Description

This is a Standard C++ Library header.

### 6.336 linear\_probe\_fn\_imp.hpp File Reference

#### 6.336.1 Detailed Description

Contains a probe policy implementation

### 6.337 list File Reference

#### Macros

- `#define _GLIBCXX_LIST`

#### 6.337.1 Detailed Description

This is a Standard C++ Library header.

### 6.338 list File Reference

#### Classes

- class `std::__debug::list<_Tp, _Allocator>`

## Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_LIST`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`

## 6.338.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.339 list File Reference

## Classes

- class [std::\\_\\_profile::list< \\_Tp, \\_Allocator >](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Macros

- `#define _GLIBCXX_PROFILE_LIST`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(*conditional *)`

### 6.339.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.340 list File Reference

### Namespaces

- [std](#)

### Macros

- `#define __GLIBCXX_EXPERIMENTAL_LIST`

### Typedefs

- `template<typename _Tp >`  
`using std::experimental::fundamentals_v2::pmr::list = std::list< _Tp, polymorphic_allocator< _Tp > >`

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >`  
`void std::experimental::fundamentals_v2::erase (list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (list< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 6.340.1 Detailed Description

This is a TS C++ Library header.

## 6.341 list.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _LIST_TCC`

#### 6.341.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 6.342 list\_partition.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter >`  
`void __gnu_parallel::__shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_↵`  
`length)`
- `template<typename _Iter >`  
`void __gnu_parallel::__shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t`  
`&__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths,`  
`const int __num_parts, _FunctorType &__f, int __oversampling=0)`

#### 6.342.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 6.343 list\_update\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::lu\\_counter\\_policy< Max\\_Count, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::lu\\_move\\_to\\_front\\_policy< \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.343.1 Detailed Description

Contains policies for list update containers.

### 6.344 locale File Reference

#### Macros

- `#define _GLIBCXX_LOCALE`

#### 6.344.1 Detailed Description

This is a Standard C++ Library header.

### 6.345 locale\_classes.h File Reference

#### Classes

- class [std::collate<\\_CharT>](#)
- class [std::collate\\_byname<\\_CharT>](#)
- class [std::locale](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)

#### Namespaces

- [std](#)

#### 6.345.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 6.346 locale\_classes.tcc File Reference

#### Namespaces

- [std](#)

## Macros

- `#define _LOCALE_CLASSES_TCC`

## Functions

- `template<typename _Facet >`  
`bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

### 6.346.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.347 locale\_conv.h File Reference

### Classes

- class [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >](#)
- class [std::wstring\\_convert< \\_Codecvt, \\_Elem, \\_Wide\\_alloc, \\_Byte\\_alloc >](#)

### Namespaces

- [std](#)

## Functions

- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn >`  
`bool std::__do_str_codecvt (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_in (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_in (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_out (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool std::__str_codecvt_out (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)`



### 6.347.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.348 locale\_facets.h File Reference

### Classes

- class [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#)
- class [std::ctype<\\_CharT>](#)
- class [std::ctype<char>](#)
- class [std::ctype<wchar\\_t>](#)
- class [std::ctype\\_byname<\\_CharT>](#)
- class [std::ctype\\_byname<char>](#)
- class [std::num\\_get<\\_CharT, \\_InIter>](#)
- class [std::num\\_put<\\_CharT, \\_OutIter>](#)
- class [std::num\\_punct<\\_CharT>](#)
- class [std::num\\_punct\\_byname<\\_CharT>](#)

### Namespaces

- [std](#)

### Macros

- `#define GLIBCXX_NUM_CXX11_FACETS`
- `#define GLIBCXX_NUM_FACETS`
- `#define GLIBCXX_NUM_UNICODE_FACETS`

### Functions

- `template<typename _CharT>`  
`_CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT * __first, const _CharT * __last)`
- `template<typename _Tp>`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT>`  
`ostreambuf_iterator<_CharT> std::__write (ostreambuf_iterator<_CharT> __s, const _CharT * __ws, int __len)`

- `template<typename _CharT, typename _OutIter >  
_OutIter std::write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT >  
bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
_CharT std::toupper (_CharT __c, const locale &__loc)`

#### 6.348.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.349 locale\_facets.tcc File Reference

### Namespaces

- `std`

### Macros

- `#define _LOCALE_FACETS_TCC`

## Functions

- `template<typename _CharT >`  
`_CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT * __first, const _CharT * __last)`
- `template<typename _CharT, typename _ValueT >`  
`int std::__int_to_char (_CharT * __bufend, _ValueT __v, const _CharT * __lit, ios_base::fmtflags __flags, bool __dec)`
- `bool std::__verify_grouping (const char * __grouping, size_t __grouping_size, const string & __grouping_tmp) throw ()`

### 6.349.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.350 locale\_facets\_nonio.h File Reference

### Classes

- class `std::messages< _CharT >`
- struct `std::messages_base`
- class `std::messages_byname< _CharT >`
- class `std::money_base`
- class `std::money_get< _CharT, _InIter >`
- class `std::money_put< _CharT, _OutIter >`
- class `std::moneypunct< _CharT, _Intl >`
- class `std::moneypunct_byname< _CharT, _Intl >`
- class `std::time_base`
- class `std::time_get< _CharT, _InIter >`
- class `std::time_get_byname< _CharT, _InIter >`
- class `std::time_put< _CharT, _OutIter >`
- class `std::time_put_byname< _CharT, _OutIter >`

### Namespaces

- `std`

### 6.350.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.351 locale\_facets\_nonio.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _LOCALE_FACETS_NONIO_TCC`

#### 6.351.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.352 localefwd.h File Reference

### Classes

- class [std::codecvt<\\_InternT, \\_ExternT, \\_StateT>](#)
- class [std::codecvt\\_byname<\\_InternT, \\_ExternT, \\_StateT>](#)
- class [std::collate<\\_CharT>](#)
- class [std::collate\\_byname<\\_CharT>](#)
- class [std::ctype<\\_CharT>](#)
- class [std::ctype\\_byname<\\_CharT>](#)
- class [std::messages<\\_CharT>](#)
- class [std::messages\\_byname<\\_CharT>](#)
- class [std::money\\_get<\\_CharT, \\_InIter>](#)
- class [std::money\\_put<\\_CharT, \\_OutIter>](#)
- class [std::moneypunct<\\_CharT, \\_Intl>](#)
- class [std::moneypunct\\_byname<\\_CharT, \\_Intl>](#)
- class [std::num\\_get<\\_CharT, \\_InIter>](#)
- class [std::num\\_put<\\_CharT, \\_OutIter>](#)
- class [std::numpunct<\\_CharT>](#)
- class [std::numpunct\\_byname<\\_CharT>](#)
- class [std::time\\_get<\\_CharT, \\_InIter>](#)
- class [std::time\\_get\\_byname<\\_CharT, \\_InIter>](#)
- class [std::time\\_put<\\_CharT, \\_OutIter>](#)
- class [std::time\\_put\\_byname<\\_CharT, \\_OutIter>](#)

### Namespaces

- [std](#)

## Functions

- `template<typename _Facet >`  
`bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet >`  
`const _Facet & std::use_facet (const locale &__loc)`

### 6.352.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 6.353 losertree.h File Reference

#### Classes

- `class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`
- `class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`
- `class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >`
- `struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`
- `class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`

- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_Loser](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >](#)

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

##### 6.353.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

## 6.354 lu\_counter\_metadata.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

##### 6.354.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

## 6.355 lu\_map\_.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

### 6.355.1 Detailed Description

Contains a list update map.

## 6.356 macros.h File Reference

### Macros

- `#define __glibcxx_check_bucket_index(_N)`
- `#define __glibcxx_check_equal_allocs(_This, _Other)`
- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_irreflexive(_First, _Last)`
- `#define __glibcxx_check_irreflexive2(_First, _Last)`
- `#define __glibcxx_check_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_check_max_load_factor(_F)`
- `#define __glibcxx_check_non_empty_range(_First, _Last)`
- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_self_move_assign(_Other)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_string(_String)`
- `#define __glibcxx_check_string_len(_String, _Len)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define __glibcxx_check_valid_range2(_First, _Last, _Dist)`
- `#define __GLIBCXX_DEBUG_VERIFY(_Condition, _ErrorMessage)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT(_Condition, _ErrorMessage, _File, _Line)`

### 6.356.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 6.356.2 Macro Definition Documentation

#### 6.356.2.1 `__glibcxx_check_erase`

```
#define __glibcxx_check_erase(  
    _Position )
```

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 145 of file macros.h.

#### 6.356.2.2 `__glibcxx_check_erase_after`

```
#define __glibcxx_check_erase_after(  
    _Position )
```

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

Definition at line 159 of file macros.h.

#### 6.356.2.3 `__glibcxx_check_erase_range`

```
#define __glibcxx_check_erase_range(  
    _First,  
    _Last )
```

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 173 of file macros.h.



**6.356.2.4 \_\_glibcxx\_check\_erase\_range\_after**

```
#define __glibcxx_check_erase_range_after(  
    _First,  
    _Last )
```

Verify that we can erase the elements in the iterator range (`_First`, `_Last`). We can erase the elements if (`_First`, `_Last`) is a valid iterator range within this sequence.

Definition at line 185 of file macros.h.

**6.356.2.5 \_\_glibcxx\_check\_heap\_pred**

```
#define __glibcxx_check_heap_pred(  
    _First,  
    _Last,  
    _Pred )
```

Verify that the iterator range [`_First`, `_Last`) is a heap w.r.t. the predicate `_Pred`.

Definition at line 335 of file macros.h.

**6.356.2.6 \_\_glibcxx\_check\_insert**

```
#define __glibcxx_check_insert(  
    _Position )
```

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 79 of file macros.h.

**6.356.2.7 \_\_glibcxx\_check\_insert\_after**

```
#define __glibcxx_check_insert_after(  
    _Position )
```

Verify that we can insert into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 96 of file macros.h.

### 6.356.2.8 \_\_glibcxx\_check\_insert\_range

```
#define __glibcxx_check_insert_range(  
    _Position,  
    _First,  
    _Last,  
    _Dist )
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

Definition at line 113 of file `macros.h`.

### 6.356.2.9 \_\_glibcxx\_check\_insert\_range\_after

```
#define __glibcxx_check_insert_range_after(  
    _Position,  
    _First,  
    _Last,  
    _Dist )
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

Definition at line 132 of file `macros.h`.

### 6.356.2.10 \_\_glibcxx\_check\_partitioned\_lower

```
#define __glibcxx_check_partitioned_lower(  
    _First,  
    _Last,  
    _Value )
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

Definition at line 279 of file `macros.h`.

**6.356.2.11 \_\_glibcxx\_check\_partitioned\_lower\_pred**

```
#define __glibcxx_check_partitioned_lower_pred(  
    _First,  
    _Last,  
    _Value,  
    _Pred )
```

Verify that the iterator range [\_First, \_Last) is partitioned w.r.t. the value \_Value and predicate \_Pred.

Definition at line 301 of file macros.h.

**6.356.2.12 \_\_glibcxx\_check\_partitioned\_upper\_pred**

```
#define __glibcxx_check_partitioned_upper_pred(  
    _First,  
    _Last,  
    _Value,  
    _Pred )
```

Verify that the iterator range [\_First, \_Last) is partitioned w.r.t. the value \_Value and predicate \_Pred.

Definition at line 314 of file macros.h.

**6.356.2.13 \_\_glibcxx\_check\_sorted\_pred**

```
#define __glibcxx_check_sorted_pred(  
    _First,  
    _Last,  
    _Pred )
```

Verify that the iterator range [\_First, \_Last) is sorted by the predicate \_Pred.

Definition at line 245 of file macros.h.

**6.356.2.14 \_GLIBCXX\_DEBUG\_VERIFY\_AT**

```
#define _GLIBCXX_DEBUG_VERIFY_AT(  
    _Condition,  
    _ErrorMessage,  
    _File,  
    _Line )
```

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 41 of file macros.h.

## 6.357 malloc\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Tp >  
bool **\_\_gnu\_cxx::operator!=** (const malloc\_allocator<\_Tp> &, const malloc\_allocator<\_Tp> &)
- template<typename \_Tp >  
bool **\_\_gnu\_cxx::operator==** (const malloc\_allocator<\_Tp> &, const malloc\_allocator<\_Tp> &)

#### 6.357.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.358 map File Reference

### Macros

- #define **\_GLIBCXX\_MAP**

#### 6.358.1 Detailed Description

This is a Standard C++ Library header.

## 6.359 map File Reference

### Macros

- #define **\_GLIBCXX\_DEBUG\_MAP**

#### 6.359.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.360 map File Reference

### Macros

- `#define _GLIBCXX_PROFILE_MAP`

### 6.360.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.361 map File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_MAP`

### Typedefs

- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::map = std::map< _Key, _Tp, _Compare, polymorphic_  
allocator< pair< const _Key, _Tp > >>>`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::multimap = std::multimap< _Key, _Tp, _Compare,  
polymorphic_allocator< pair< const _Key, _Tp > >>>`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate >  
void std::experimental::fundamentals_v2::erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _  
Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate >  
void std::experimental::fundamentals_v2::erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _  
Predicate __pred)`

### 6.361.1 Detailed Description

This is a TS C++ Library header.

## 6.362 map.h File Reference

## Classes

- class [std::\\_\\_debug::map<\\_Key, \\_Tp, \\_Compare, \\_Allocator>](#)

## Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator< (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator<= (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _Allocator> &__lhs, const map< _Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator> &__lhs, map< _Key, _Tp, _Compare, _Allocator> &__rhs) noexcept(/*conditional */)`

## 6.362.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.363 map.h File Reference

## Classes

- class [std::\\_\\_profile::map<\\_Key, \\_Tp, \\_Compare, \\_Allocator>](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
void std::__profile::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

### 6.363.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

### 6.364 mask\_array.h File Reference

#### Classes

- class [std::mask\\_array<\\_Tp>](#)

#### Namespaces

- [std](#)

#### Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 6.364.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

6.365 `mask_based_range_hashing.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 6.365.1 Detailed Description

Contains a range hashing policy base.

6.366 `math.h` File Reference

## 6.366.1 Detailed Description

This is a Standard C++ Library header.

6.367 `memory` File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_MEMORY`

## Enumerations

- enum `pointer_safety` { `relaxed`, `preferred`, `strict` }



## Functions

- void \* [std::align](#) (size\_t \_\_align, size\_t \_\_size, void \*&\_\_ptr, size\_t &\_\_space) noexcept
- void **std::declare\_no\_pointers** (char \*, size\_t)
- void **std::declare\_reachable** (void \*)
- pointer\_safety **std::get\_pointer\_safety** () noexcept
- void **std::undeclare\_no\_pointers** (char \*, size\_t)
- template<typename \_Tp >  
\_Tp \* **std::undeclare\_reachable** (\_Tp \*\_\_p)

### 6.367.1 Detailed Description

This is a Standard C++ Library header.

## 6.368 memory File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::temporary\\_buffer](#)< [\\_ForwardIterator](#), [\\_Tp](#) >

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- #define **\_EXT\_MEMORY**

### Functions

- template<typename [\\_InputIter](#) , typename [\\_Size](#) , typename [\\_ForwardIter](#) >  
[pair](#)< [\\_InputIter](#) , [\\_ForwardIter](#) > **\_\_gnu\_cxx::uninitialized\_copy\_n** ([\\_InputIter](#) \_\_first, [\\_Size](#) \_\_count, [\\_ForwardIter](#) \_\_result, [std::input\\_iterator\\_tag](#))
- template<typename [\\_RandomAccessIter](#) , typename [\\_Size](#) , typename [\\_ForwardIter](#) >  
[pair](#)< [\\_RandomAccessIter](#) , [\\_ForwardIter](#) > **\_\_gnu\_cxx::uninitialized\_copy\_n** ([\\_RandomAccessIter](#) \_\_first, [\\_Size](#) \_\_count, [\\_ForwardIter](#) \_\_result, [std::random\\_access\\_iterator\\_tag](#))
- template<typename [\\_InputIter](#) , typename [\\_Size](#) , typename [\\_ForwardIter](#) >  
[pair](#)< [\\_InputIter](#) , [\\_ForwardIter](#) > **\_\_gnu\_cxx::uninitialized\_copy\_n** ([\\_InputIter](#) \_\_first, [\\_Size](#) \_\_count, [\\_ForwardIter](#) \_\_result)
- template<typename [\\_InputIter](#) , typename [\\_Size](#) , typename [\\_ForwardIter](#) , typename [\\_Allocator](#) >  
[pair](#)< [\\_InputIter](#) , [\\_ForwardIter](#) > **\_\_gnu\_cxx::uninitialized\_copy\_n\_a** ([\\_InputIter](#) \_\_first, [\\_Size](#) \_\_count, [\\_ForwardIter](#) \_\_result, [\\_Allocator](#) \_\_alloc)
- template<typename [\\_InputIter](#) , typename [\\_Size](#) , typename [\\_ForwardIter](#) , typename [\\_Tp](#) >  
[pair](#)< [\\_InputIter](#) , [\\_ForwardIter](#) > **\_\_gnu\_cxx::uninitialized\_copy\_n\_a** ([\\_InputIter](#) \_\_first, [\\_Size](#) \_\_count, [\\_ForwardIter](#) \_\_result, [std::allocator](#)< [\\_Tp](#) >)
- template<typename [\\_InputIter](#) , typename [\\_Size](#) , typename [\\_ForwardIter](#) >  
[pair](#)< [\\_InputIter](#) , [\\_ForwardIter](#) > [\\_\\_gnu\\_cxx::uninitialized\\_copy\\_n](#) ([\\_InputIter](#) \_\_first, [\\_Size](#) \_\_count, [\\_ForwardIter](#) \_\_result)

## 6.368.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.369 memory File Reference

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_experimental_observer_ptr`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY`

## Functions

- `template<typename _Tp >`  
`observer_ptr< _Tp > std::experimental::fundamentals_v2::make_observer (_Tp *__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator!= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator== (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::swap (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept`

### 6.369.1 Detailed Description

This is a TS C++ Library header.

## 6.370 memory\_resource File Reference

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_experimental_memory_resources`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY_RESOURCE`

### Typedefs

- `template<typename _Alloc >`  
using **std::experimental::fundamentals\_v2::pmr::resource\_adaptor** = \_\_resource\_adaptor\_imp< typename allocator\_traits< \_Alloc >::template rebind\_alloc< char > >

### Functions

- `std::atomic< memory_resource * > & std::experimental::fundamentals_v2::pmr::__get_default_resource()`
- `memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource () noexcept`
- `memory_resource * std::experimental::fundamentals_v2::pmr::new_delete_resource () noexcept`
- `memory_resource * std::experimental::fundamentals_v2::pmr::null_memory_resource () noexcept`
- `bool std::experimental::fundamentals_v2::pmr::operator!= (const memory_resource &__a, const memory_resource &__b) noexcept`
- `template<class _Tp1 , class _Tp2 >`  
`bool std::experimental::fundamentals_v2::pmr::operator!= (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept`
- `bool std::experimental::fundamentals_v2::pmr::operator== (const memory_resource &__a, const memory_resource &__b) noexcept`
- `template<class _Tp1 , class _Tp2 >`  
`bool std::experimental::fundamentals_v2::pmr::operator== (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept`
- `memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (memory_resource *__r) noexcept`

### 6.370.1 Detailed Description

This is a TS C++ Library header.

## 6.371 memoryfwd.h File Reference

## Classes

- class [std::allocator<\\_Tp>](#)
- struct [std::uses\\_allocator<\\_Tp, \\_Alloc>](#)

## Namespaces

- [std](#)

## 6.371.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.372 merge.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1>::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1>::difference_type __max_length, _Compare __comp)`

## 6.372.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.373 `messages_members.h` File Reference

### Namespaces

- [std](#)

### 6.373.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.374 `mod_based_range_hashing.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_Type >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.374.1 Detailed Description

Contains a range hashing policy base.

## 6.375 `move.h` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_FORWARD(__Tp, __val)`
- `#define _GLIBCXX_MOVE(__val)`

## Functions

- `template<typename _Tp >`  
`constexpr _Tp * std::\_\_addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`  
`_Tp std::\_\_exchange (_Tp &__obj, _Up &&__new_val)`
- `template<typename _Tp >`  
`_GLIBCXX17_CONSTEXPR _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`  
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr std::remove\_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type std::move\_if\_noexcept (_Tp &__x) noexcept`
- `template<typename _Tp >`  
`enable_if< __and< __not< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > >::value >::type std::swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm>`  
`enable_if< __is_swappable< _Tp >::value >::type std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)`

## 6.375.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 6.376 mt\_allocator.h File Reference

## Classes

- `struct \_\_gnu\_cxx::\_\_common\_pool\_policy< _PoolTp, _Thread >`
- `class \_\_gnu\_cxx::\_\_mt\_alloc< _Tp, _Poolp >`
- `class \_\_gnu\_cxx::\_\_mt\_alloc\_base< _Tp >`
- `struct \_\_gnu\_cxx::\_\_per\_type\_pool\_policy< _Tp, _PoolTp, _Thread >`
- `class \_\_gnu\_cxx::\_\_pool< _Thread >`
- `class \_\_gnu\_cxx::\_\_pool< false >`
- `class \_\_gnu\_cxx::\_\_pool< true >`
- `struct \_\_gnu\_cxx::\_\_pool\_base`

## Namespaces

- `\_\_gnu\_cxx`

## Macros

- `#define __thread_default`

## Typedefs

- `typedef void(* __gnu_cxx::__destroy_handler) (void *)`

## Functions

- `template<typename _Tp, typename _Poolp >  
bool __gnu_cxx::operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp, typename _Poolp >  
bool __gnu_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`

### 6.376.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 6.377 multimap.h File Reference

#### Classes

- class `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`

#### Namespaces

- `std`
- `std::__debug`

#### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

## 6.377.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.378 multimap.h File Reference

## Classes

- class [std::\\_\\_profile::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__profile::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
void std::__profile::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

## 6.378.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.



## 6.379 multiseq\_selection.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_Lexicographic<\\_T1, \\_T2, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LexicographicReverse<\\_T1, \\_T2, \\_Compare>](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- `#define __S(__i)`
- `#define __S(__i)`

### Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void \_\_gnu\_parallel::multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`  
`_RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::`  
`::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp \_\_gnu\_parallel::multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`  
`_RankType &__offset, _Compare __comp=std::less< _Tp >())`

### 6.379.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 12(2):171177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

## 6.380 multiset.h File Reference

### Classes

- class `std::__debug::multiset<_Key, _Compare, _Allocator>`

## Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)`

## 6.380.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.381 multiset.h File Reference

## Classes

- class [std::\\_\\_profile::multiset< \\_Key, \\_Compare, \\_Allocator >](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)`

### 6.381.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.382 multiway\_merge.h File Reference

### Classes

- `struct \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- `struct \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- `struct \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- `struct \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- `struct \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- `struct \_\_gnu\_parallel::\_\_multiway\_merge\_k\_variant\_sentinel\_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- `class \_\_gnu\_parallel::GuardedIterator< _RAIter, _Compare >`
- `struct \_\_gnu\_parallel::LoserTreeTraits< _Tp >`
- `struct \_\_gnu\_parallel::SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`
- `struct \_\_gnu\_parallel::SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

### Namespaces

- `\_\_gnu\_parallel`

## Macros

- `#define _GLIBCXX_PARALLEL_DECISION(__a, __b, __c, __d)`
- `#define _GLIBCXX_PARALLEL_LENGTH(__s)`
- `#define _GLIBCXX_PARALLEL_MERGE_3_CASE(__a, __b, __c, __c0, __c1)`
- `#define _GLIBCXX_PARALLEL_MERGE_4_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >  
_OutputIterator \_\_gnu\_parallel::merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
<_RAIter3 \_\_gnu\_parallel::sequential\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag\(0\))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >  
void \_\_gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< \_DifferenceType, \_DifferenceType > > \* \_\_pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator`  
`__seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIter`  
`Iterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void \_\_gnu\_parallel::multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator`  
`__seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector<`  
`std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel`  
`tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter,`  
`typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,`  
`_RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num`  
`threads)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel`  
`tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIter`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIter`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag`  
`tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`

### 6.382.1 Detailed Description

Implementation of sequential and parallel multiway merge.

Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

### 6.382.2 Macro Definition Documentation

#### 6.382.2.1 \_GLIBCXX\_PARALLEL\_LENGTH

```
#define _GLIBCXX_PARALLEL_LENGTH(  
    __s )
```

Length of a sequence described by a pair of iterators.

Definition at line 54 of file multiway\_merge.h.

## 6.383 multiway\_mergesort.h File Reference

### Classes

- `struct gnu\_parallel::Piece< _DifferenceTp >`
- `struct gnu\_parallel::PMWMSortingData< _RAIter >`
- `struct gnu\_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >`
- `struct gnu\_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`
- `struct gnu\_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`

### Namespaces

- [gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp >`  
`void \_\_gnu\_parallel::\_\_determine\_samples (_PMWMSSortingData< _RAIter > *__sd, _DifferenceTp __num↵  
_samples)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex  
__num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::parallel\_sort\_mwms\_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)`

### 6.383.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

## 6.384 mutex File Reference

### Classes

- struct [std::once\\_flag](#)
- class [std::recursive\\_mutex](#)
- class [std::recursive\\_timed\\_mutex](#)
- class [std::timed\\_mutex](#)

### Namespaces

- [std](#)

### Macros

- `#define GLIBCXX\_MUTEX`
- `__thread void * std::\_\_once\_callable`
- `__thread void(* std::\_\_once\_call )()`
- `template<typename _Lock >`  
`unique_lock< _Lock > std::\_\_try\_to\_lock (_Lock &__l)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`  
`int std::try\_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &... __l3)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)`
- `void std::\_\_once\_proxy (void)`
- `template<typename _Callable, typename... _Args>`  
`void std::call\_once (once_flag &__once, _Callable &&__f, _Args &&... __args)`

## 6.384.1 Detailed Description

This is a Standard C++ Library header.

## 6.385 nested\_exception.h File Reference

## Classes

- class [std::nested\\_exception](#)

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Tp >`  
using **std::\_\_rethrow\_if\_nested\_cond** = typename enable\_if< \_\_and< is\_polymorphic< \_Tp >, \_\_or<   
\_\_not< is\_base\_of< nested\_exception, \_Tp >, is\_convertible< \_Tp \*, nested\_exception \* > > >::value  
>::type

## Functions

- `template<typename _Ex >`  
`__rethrow_if_nested_cond< _Ex > std::__rethrow_if_nested_impl (const _Ex *__ptr)`
- `void std::__rethrow_if_nested_impl (const void *)`
- `template<typename _Tp >`  
`void std::__throw_with_nested_impl (_Tp &&__t, true_type)`
- `template<typename _Tp >`  
`void std::__throw_with_nested_impl (_Tp &&__t, false_type)`
- `template<typename _Ex >`  
`void std::rethrow\_if\_nested (const _Ex &__ex)`
- `template<typename _Tp >`  
`void std::throw\_with\_nested (_Tp &&__t)`

## 6.385.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 6.386 new File Reference

## Classes

- class [std::bad\\_alloc](#)



## Namespaces

- [std](#)

## Typedefs

- typedef void(\* [std::new\\_handler](#)) ()

## Functions

- new\_handler [std::get\\_new\\_handler](#) () noexcept
- new\_handler [std::set\\_new\\_handler](#) (new\_handler) throw ()
  
- void \* [operator new](#) (std::size\_t) \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void \* [operator new\[\]](#) (std::size\_t) \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void [operator delete](#) (void \*) noexcept \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void [operator delete\[\]](#) (void \*) noexcept \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void \* [operator new](#) (std::size\_t, const std::nothrow\_t &) noexcept \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void \* [operator new\[\]](#) (std::size\_t, const std::nothrow\_t &) noexcept \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void [operator delete](#) (void \*, const std::nothrow\_t &) noexcept \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void [operator delete\[\]](#) (void \*, const std::nothrow\_t &) noexcept \_\_attribute\_\_((\_\_externally\_visible\_\_))
- void \* [operator new](#) (std::size\_t, void \*\_\_p) noexcept
- void \* [operator new\[\]](#) (std::size\_t, void \*\_\_p) noexcept
- void [operator delete](#) (void \*, void \*) noexcept
- void [operator delete\[\]](#) (void \*, void \*) noexcept

## Variables

- const nothrow\_t **std::nothrow**

### 6.386.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see [http://gcc.gnu.org/onlinedocs/libstdc++/18\\_support/howto.html#4](http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4) for more.

### 6.386.2 Function Documentation

**6.386.2.1 operator delete()** [1/3]

```
void operator delete (  
    void * ) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.2 operator delete()** [2/3]

```
void operator delete (  
    void * ,  
    const std::nothrow_t & ) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.3 operator delete()** [3/3]

```
void operator delete (  
    void * ,  
    void * ) [inline], [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 174 of file `new`.

**6.386.2.4 operator delete[]()** [1/3]

```
void operator delete[] (
    void * ) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.5 operator delete[]()** [2/3]

```
void operator delete[] (
    void * ,
    const std::nothrow_t & ) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.6 operator delete[]()** [3/3]

```
void operator delete[] (
    void * ,
    void * ) [inline], [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 175 of file new.

**6.386.2.7 operator new()** [1/3]

```
void* operator new (
    std::size_t )
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.8 operator new()** [2/3]

```
void* operator new (
    std::size_t ,
    const std::nothrow_t & ) [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.9 operator new()** [3/3]

```
void* operator new (
    std::size_t ,
    void * __p ) [inline], [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 168 of file new.

**6.386.2.10** `operator new[]()` [1/3]

```
void* operator new[] (
    std::size_t )
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.11** `operator new[]()` [2/3]

```
void* operator new[] (
    std::size_t ,
    const std::nothrow_t & ) [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.386.2.12** `operator new[]()` [3/3]

```
void* operator new[] (
    std::size_t ,
    void * __p ) [inline], [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 170 of file `new`.

## 6.387 new\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::new\\_allocator<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp>  
bool __gnu_cxx::operator!= (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`
- `template<typename _Tp>  
bool __gnu_cxx::operator== (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`

#### 6.387.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.388 node.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_<\\_Value, \\_Metadata, \\_Alloc>](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.388.1 Detailed Description

Contains an implementation struct for this type of heap's node.

## 6.389 node.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_node\\_<Value\\_Type, Metadata, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.389.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 6.390 `node.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_node\\_](#)< `Value_Type`, `Metadata`, `_Alloc` >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.390.1 Detailed Description

Contains an implementation struct for `splay_tree_`'s node.

## 6.391 `node_handle.h` File Reference

### 6.391.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map,set,unordered_map,unordered_set>`.

## 6.392 `node_iterators.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it\\_](#)< `Node`, `Const_Iterator`, `Iterator`, `_Alloc` >
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it\\_](#)< `Node`, `Const_Iterator`, `Iterator`, `_Alloc` >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC`
- `#define PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC`

## 6.392.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.393 node\_iterators.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_const\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC`
- `#define PB_DS_OV_TREE_NODE_ITERATOR_C_DEC`

## 6.393.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 6.394 node\_metadata\_selector.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, \\_BTp >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)



#### 6.394.1 Detailed Description

Contains an implementation class for trees.

### 6.395 `node_metadata_selector.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, \\_BTp >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.395.1 Detailed Description

Contains an implementation class for tries.

### 6.396 `null_node_metadata.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator< Key, Data, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.396.1 Detailed Description

Contains an implementation class for tree-like classes.

### 6.397 `numeric` File Reference

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Macros

- `#define _GLIBCXX_NUMERIC`

## Functions

- `template<typename _Up, typename _Tp >`  
`constexpr _Up std::__detail::__absu (_Tp __val)`
- `template<typename _Up >`  
`void std::__detail::__absu (bool)=delete`
- `template<typename _Tp >`  
`constexpr _Tp std::__detail::__gcd (_Tp __m, _Tp __n)`
- `template<typename _Tp >`  
`constexpr _Tp std::__detail::__lcm (_Tp __m, _Tp __n)`

### 6.397.1 Detailed Description

This is a Standard C++ Library header.

## 6.398 numeric File Reference

## Namespaces

- [`\_\_gnu\_cxx`](#)

## Macros

- `#define _EXT_NUMERIC`

## Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`

### 6.398.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 6.399 numeric File Reference

### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

### Macros

- `#define GLIBCXX_PARALLEL_NUMERIC_H`

### Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate_switch (_RAIter __begin, _RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Tp std::__parallel::__inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op,`  
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`__gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, ↵`  
`_BinaryOperation __bin_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, ↵`  
`_BinaryOperation __binary_op, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, ↵`  
`_BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::__Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2,`  
`__gnu_parallel::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _Binary↵`  
`Function1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`__gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary↵`  
`Operation __bin_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary↵`  
`Operation __binary_op)`

### 6.399.1 Detailed Description

Parallel STL function calls corresponding to `stl_numeric.h`. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 6.400 numeric File Reference

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_experimental_gcd_lcm`
- `#define _GLIBCXX_EXPERIMENTAL_NUMERIC`

### Functions

- `template<typename _Mn, typename _Nn >`  
`constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals\_v2::gcd (_Mn __m, _Nn __n) noexcept`
- `template<typename _Mn, typename _Nn >`  
`constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals\_v2::lcm (_Mn __m, _Nn __n)`

#### 6.400.1 Detailed Description

This is a TS C++ Library header.

#### 6.400.2 Function Documentation

##### 6.400.2.1 gcd()

```
template<typename _Mn, typename _Nn >
constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals_v2::gcd (
    _Mn __m,
    _Nn __n ) [noexcept]
```

Greatest common divisor.

Definition at line 56 of file experimental/numeric.

##### 6.400.2.2 lcm()

```
template<typename _Mn, typename _Nn >
constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals_v2::lcm (
    _Mn __m,
    _Nn __n )
```

Least common multiple.

Definition at line 74 of file experimental/numeric.

## 6.401 numeric\_traits.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define __glibcxx_digits(_Tp)`
- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max(_Tp)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __glibcxx_min(_Tp)`
- `#define __glibcxx_signed(_Tp)`

## 6.401.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.402 numericfwd.h File Reference

## Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

## Functions

- `template<typename _Iter, typename _Tp, typename _Tag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag,`  
`\_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag,`  
`random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2,`  
`random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >`  
`_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::__partial_sum (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::__partial_sum (_Iter, _Iter, _OIter __result)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >  
_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`

#### 6.402.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

### 6.403 omp\_loop.h File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op <↵  
__o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter ><↵  
::difference_type __bound)`

#### 6.403.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

### 6.404 omp\_loop\_static.h File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end,  
_Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter ><↵  
::difference_type __bound)`

#### 6.404.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.



## 6.405 `opt_random.h` File Reference

### Namespaces

- [std](#)

### 6.405.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.406 `optional` File Reference

### Classes

- struct [std::experimental::fundamentals\\_v1::\\_Has\\_addressof<\\_Tp>](#)
- class [std::experimental::fundamentals\\_v1::\\_Optional\\_base<\\_Tp, \\_ShouldProvideDestructor>](#)
- class [std::experimental::fundamentals\\_v1::\\_Optional\\_base<\\_Tp, false>](#)
- class [std::experimental::fundamentals\\_v1::bad\\_optional\\_access](#)
- struct [std::experimental::fundamentals\\_v1::in\\_place\\_t](#)
- struct [std::experimental::fundamentals\\_v1::nullopt\\_t](#)
- class [std::experimental::fundamentals\\_v1::optional<\\_Tp>](#)
- class [std::experimental::fundamentals\\_v1::optional<\\_Tp>](#)
- class [std::experimental::fundamentals\\_v1::optional<\\_Tp>](#)

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_experimental_optional`
- `#define _GLIBCXX_EXPERIMENTAL_OPTIONAL`

### Typedefs

- `template<typename _Tp, typename _Up>`  
using **`std::experimental::fundamentals_v1::_assigns_from_optional`** = `__or_< is_assignable<_Tp &, const optional<_Up> &>, is_assignable<_Tp &, optional<_Up> &>, is_assignable<_Tp &, const optional<_Up> &&>, is_assignable<_Tp &, optional<_Up> &&>>`
- `template<typename _Tp, typename _Up>`  
using **`std::experimental::fundamentals_v1::_converts_from_optional`** = `__or_< is_constructible<_Tp, const optional<_Up> &>, is_constructible<_Tp, optional<_Up> &>, is_constructible<_Tp, const optional<_Up> &&>, is_constructible<_Tp, optional<_Up> &&>, is_convertible<const optional<_Up> &, _Tp>, is_convertible<optional<_Up> &, _Tp>, is_convertible<const optional<_Up> &&, _Tp>, is_convertible<optional<_Up> &&, _Tp>>`

## Functions

- `template<typename _Tp >`  
`constexpr enable_if_t<!_Has_addressof<_Tp>::value, _Tp*> std::experimental::fundamentals\_v1::\_\_constexpr\_addressof`  
`(_Tp &__t)`
- `template<typename _Tp >`  
`enable_if_t<_Has_addressof<_Tp>::value, _Tp*> std::experimental::fundamentals\_v1::\_\_constexpr\_addressof`  
`(_Tp &__t)`
- `void std::experimental::fundamentals_v1::__throw_bad_optional_access (const char *) __attribute__((__noreturn__))`
- `template<typename _Tp >`  
`constexpr optional<decay_t<_Tp>> std::experimental::fundamentals_v1::make_optional (_Tp &&__t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const optional<_Tp> &__lhs, const optional<_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const optional<_Tp> &__lhs, nullopt_t __t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (nullopt_t, const optional<_Tp> &__rhs) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const optional<_Tp> &__lhs, _Tp const &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (const _Tp &__lhs, const optional<_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const optional<_Tp> &__lhs, const optional<_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const optional<_Tp> &, nullopt_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (nullopt_t, const optional<_Tp> &__rhs) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const optional<_Tp> &__lhs, const _Tp &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (const _Tp &__lhs, const optional<_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const optional<_Tp> &__lhs, const optional<_Tp> &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const optional<_Tp> &__lhs, nullopt_t) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (nullopt_t, const optional<_Tp> &) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const optional<_Tp> &__lhs, const _Tp &__rhs)`

- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (const _Tp &__lhs, const optional< _Tp >`  
`&__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const optional< _Tp > &__lhs, const`  
`optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const optional< _Tp > &__lhs, nullopt_t)`  
`noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (nullopt_t, const optional< _Tp > &__rhs)`  
`noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const optional< _Tp > &__lhs, const _Tp`  
`&__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (const _Tp &__lhs, const optional< _Tp >`  
`&__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const optional< _Tp > &__lhs, const`  
`optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const optional< _Tp > &__lhs, nullopt_t`  
`) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (nullopt_t, const optional< _Tp > &) noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const optional< _Tp > &__lhs, const _Tp`  
`&__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (const _Tp &__lhs, const optional< _Tp >`  
`&__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const optional< _Tp > &__lhs, const`  
`optional< _Tp > &__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const optional< _Tp > &, nullopt_t) noex-`  
`cept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (nullopt_t, const optional< _Tp > &__rhs)`  
`noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const optional< _Tp > &__lhs, const _Tp`  
`&__rhs)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (const _Tp &__lhs, const optional< _Tp >`  
`&__rhs)`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v1::swap (optional< _Tp > &__lhs, optional< _Tp > &__rhs)`  
`noexcept(noexcept(__lhs.swap(__rhs)))`

## Variables

- constexpr in\_place\_t [std::experimental::fundamentals\\_v1::in\\_place](#)
- constexpr nullopt\_t [std::experimental::fundamentals\\_v1::nullopt](#)

### 6.406.1 Detailed Description

This is a TS C++ Library header.

### 6.406.2 Function Documentation

#### 6.406.2.1 \_\_constexpr\_addressof()

```
template<typename _Tp >
enable_if_t<_Has_addressof<_Tp>::value, _Tp*> std::experimental::fundamentals_v1::__constexpr_↵
addressof (
    _Tp & __t ) [inline]
```

Fallback overload that defers to \_\_addressof.

Definition at line 184 of file optional.

## 6.407 order\_statistics\_imp.hpp File Reference

### 6.407.1 Detailed Description

Contains forward declarations for order\_statistics\_key

## 6.408 order\_statistics\_imp.hpp File Reference

### 6.408.1 Detailed Description

Contains forward declarations for order\_statistics\_key

## 6.409 ordered\_base.h File Reference

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

#### 6.409.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

### 6.410 `os_defines.h` File Reference

#### Macros

- `#define __NO_CTYPE`
- `#define _GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC`

#### 6.410.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

### 6.411 `ostream` File Reference

#### Classes

- class `std::basic_ostream<_CharT, _Traits>`
- class `std::basic_ostream<_CharT, _Traits>::sentry`

#### Namespaces

- `std`

#### Macros

- `#define _GLIBCXX_OSTREAM`

#### Typedefs

- `template<typename _Tp>`  
using `std::__do_is_convertible_to_basic_ostream_impl` = `decltype(__is_convertible_to_basic_ostream_↵  
test(declval<typename remove_reference<_Tp>::type*>()))`
- `template<typename _Ostream>`  
using `std::__rvalue_ostream_type` = `typename __is_convertible_to_basic_ostream<_Ostream>::__↵  
ostream_type`

## Functions

- `template<typename _Ch, typename _Up >`  
`basic_ostream< _Ch, _Up > & std::__is_convertible_to_basic_ostream_test (basic_ostream< _Ch, _Up >`  
`*)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::endl (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::ends (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::flush (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _Ostream, typename _Tp >`  
`enable_if< __and< __not< is_lvalue_reference< _Ostream >, __is_convertible_to_basic_ostream< ↵`  
`_Ostream >, __is_insertable< __rvalue_ostream_type< _Ostream >, const _Tp & >::value, __rvalue_↵`  
`ostream_type< _Ostream > >::type std::operator<< (_Ostream &&__os, const _Tp &__x)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT`  
`*__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char`  
`*__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char`  
`*__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned`  
`char *__s)`

## 6.411.1 Detailed Description

This is a Standard C++ Library header.

## 6.412 ostream.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _OSTREAM_TCC`

### Functions

- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`

#### 6.412.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 6.413 ostream\_insert.h File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _CharT, typename _Traits >  
void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >  
void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`

#### 6.413.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 6.414 ov\_tree\_map.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >::cond\\_dtor< Size\\_Type >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_NODE_ITERATOR_NAME`
- `#define PB_DS_OV_TREE_NAME`
- `#define PB_DS_OV_TREE_TRAITS_BASE`

#### 6.414.1 Detailed Description

Contains an implementation class for ov\_tree.

## 6.415 pairing\_heap.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pairing\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_P_HEAP_BASE`

#### 6.415.1 Detailed Description

Contains an implementation class for a pairing heap.



## 6.416 par\_loop.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu  
&__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type  
__bound)`

#### 6.416.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## 6.417 parallel.h File Reference

#### 6.417.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

## 6.418 parse\_numbers.h File Reference

### Namespaces

- [std](#)

### Typedefs

- `template<unsigned long long _Val>  
using std::__parse_int::__ull_constant = integral_constant< unsigned long long, _Val >`
- `template<char... _Digs>  
using std::__select_int::__Select_int = typename _Select_int_base< __parse_int::__Parse_int< _Digs... >↔  
::value, unsigned char, unsigned short, unsigned int, unsigned long, unsigned long long >::type`

#### 6.418.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

## 6.419 `partial_sum.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result, ↵`  
`_BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator`  
`__result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __↵`  
`result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::difference_type __n)`

#### 6.419.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

## 6.420 `partition.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_VOLATILE`

### Functions

- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __↵`  
`comp)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __↵`  
`comp)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_partition (_RAIter __begin, _RAIter ↵`  
`__end, _Predicate __pred, _ThreadIndex __num_threads)`

#### 6.420.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

#### 6.420.2 Macro Definition Documentation

##### 6.420.2.1 `_GLIBCXX_VOLATILE`

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variables volatile.

Definition at line 43 of file `partition.h`.

#### 6.421 `pat_trie_.hpp` File Reference

##### Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

##### Namespaces

- [\\_\\_gnu\\_pbds](#)

##### Macros

- `#define PB_DS_ASSERT_NODE_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_PAT_TRIE_NAME`
- `#define PB_DS_PAT_TRIE_TRAITS_BASE`
- `#define PB_DS_RECURSIVE_COUNT_LEAFS(X)`

##### 6.421.1 Detailed Description

Contains an implementation class for a patricia tree.

## 6.422 pat\_trie\_base.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::const\\_iterator](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::iterator](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Metadata, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base< \\_ATraits, Metadata >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_IT_C_DEC`
- `#define PB_DS_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_IT_C_DEC`
- `#define PB_DS_ODIR_IT_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC`

## 6.422.1 Detailed Description

Contains the base class for a patricia tree.

## 6.423 pod\_char\_traits.h File Reference

## Classes

- struct [\\_\\_gnu\\_cxx::character< \\_Value, \\_Int, \\_St >](#)
- struct [std::char\\_traits< \\_\\_gnu\\_cxx::character< \\_Value, \\_Int, \\_St > >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Functions

- `template<typename _Value, typename _Int, typename _St >`  
`bool __gnu_cxx::operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St >`  
`&rhs)`
- `template<typename _Value, typename _Int, typename _St >`  
`bool __gnu_cxx::operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St >`  
`&rhs)`

### 6.423.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.424 point\_const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.424.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 6.425 point\_const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_< Node, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 6.425.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

6.426 `point_const_iterator.hpp` File Reference

## Classes

- class [point\\_const\\_iterator\\_](#)

## 6.426.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

6.427 `point_iterator.hpp` File Reference

## Classes

- class [point\\_iterator\\_](#)

## 6.427.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

6.428 `point_iterators.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

### 6.428.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.429 pointer.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Invalid\\_type](#)
- class [\\_\\_gnu\\_cxx::Pointer\\_adapter<\\_Storage\\_policy>](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl<const \\_Tp>](#)
- class [\\_\\_gnu\\_cxx::Std\\_pointer\\_impl<\\_Tp>](#)
- struct [\\_\\_gnu\\_cxx::Unqualified\\_type<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

### Functions

- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter<_Tp1> &__lhs, const _Pointer_adapter<_Tp2> &__lhs, const _Pointer_adapter<_Tp2> &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter<_Tp> &__lhs, int __rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter<_Tp> &__rhs)`

- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`



#### 6.429.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

##### Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

#### 6.430 `policy_access_fn_imps.hpp` File Reference

##### 6.430.1 Detailed Description

Contains an implementation class for a `binary_heap`.

#### 6.431 `policy_access_fn_imps.hpp` File Reference

##### 6.431.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

#### 6.432 `policy_access_fn_imps.hpp` File Reference

##### 6.432.1 Detailed Description

Contains implementations of `cc_ht_map_`'s policy access functions.

#### 6.433 `policy_access_fn_imps.hpp` File Reference

##### 6.433.1 Detailed Description

Contains implementations of `gp_ht_map_`'s policy access functions.

#### 6.434 `policy_access_fn_imps.hpp` File Reference

##### 6.434.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.435 policy\_access\_fn\_imps.hpp File Reference

### 6.435.1 Detailed Description

Contains an implementation class for `ov_tree`.

## 6.436 policy\_access\_fn\_imps.hpp File Reference

### 6.436.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.437 pool\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc\\_base](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp>  
bool __gnu_cxx::operator!= (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`
- `template<typename _Tp>  
bool __gnu_cxx::operator== (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`

### 6.437.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.438 postypes.h File Reference

### Classes

- class [std::fpos<\\_StateT>](#)

## Namespaces

- [std](#)

## Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate\_t > [std::streampos](#)
- typedef ptrdiff\_t [std::streamsize](#)
- typedef fpos< mbstate\_t > [std::u16streampos](#)
- typedef fpos< mbstate\_t > [std::u32streampos](#)
- typedef fpos< mbstate\_t > [std::wstreampos](#)

## Functions

- template<typename \_StateT >  
bool **std::operator!=** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)
- template<typename \_StateT >  
bool [std::operator==](#) (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)

### 6.438.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 6.439 predefined\_ops.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- template<typename \_Compare >  
[\\_\\_GLIBCXX14\\_CONSTEXPR](#) \_Iter\_comp\_iter< \_Compare > [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_iter](#) (\_Compare \_\_comp)
- template<typename \_Iterator >  
[\\_Iter\\_equals\\_iter](#)< \_Iterator > [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_iter](#) (\_Iter\_equal\_to\_iter, \_Iterator \_\_it)
- template<typename \_Compare, typename \_Iterator >  
[\\_Iter\\_comp\\_to\\_iter](#)< \_Compare, \_Iterator > [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_iter](#) (\_Iter\_comp\_iter< \_↔  
\_Compare > \_\_comp, \_Iterator \_\_it)
- [\\_Iter\\_less\\_val](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_val](#) (\_Iter\_less\_iter)
- [\\_Iter\\_equal\\_to\\_val](#) [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_val](#) (\_Iter\_equal\_to\_iter)
- template<typename \_Compare >  
[\\_Iter\\_comp\\_val](#)< \_Compare > [\\_\\_gnu\\_cxx::\\_\\_ops::\\_\\_iter\\_comp\\_val](#) (\_Compare \_\_comp)

- `template<typename _Compare >`  
`_Iter_comp_val< _Compare > __gnu_cxx::__ops::__iter_comp_val (_Iter_comp_iter< _Compare > __comp)`
- `template<typename _Compare, typename _Value >`  
`_Iter_comp_to_val< _Compare, _Value > __gnu_cxx::__ops::__iter_comp_val (_Compare __comp, _Value &__val)`
- `_Iter_equal_to_iter __gnu_cxx::__ops::__iter_equal_to_iter ()`
- `_Iter_equal_to_val __gnu_cxx::__ops::__iter_equal_to_val ()`
- `template<typename _Value >`  
`_Iter_equals_val< _Value > __gnu_cxx::__ops::__iter_equals_val (_Value &__val)`
- `_GLIBCXX14_CONSTEXPR _Iter_less_iter __gnu_cxx::__ops::__iter_less_iter ()`
- `_Iter_less_val __gnu_cxx::__ops::__iter_less_val ()`
- `template<typename _Predicate >`  
`_Iter_negate< _Predicate > __gnu_cxx::__ops::__negate (_Iter_pred< _Predicate > __pred)`
- `template<typename _Predicate >`  
`_Iter_pred< _Predicate > __gnu_cxx::__ops::__pred_iter (_Predicate __pred)`
- `_Val_less_iter __gnu_cxx::__ops::__val_comp_iter (_Iter_less_iter)`
- `template<typename _Compare >`  
`_Val_comp_iter< _Compare > __gnu_cxx::__ops::__val_comp_iter (_Compare __comp)`
- `template<typename _Compare >`  
`_Val_comp_iter< _Compare > __gnu_cxx::__ops::__val_comp_iter (_Iter_comp_iter< _Compare > __comp)`
- `_Val_less_iter __gnu_cxx::__ops::__val_less_iter ()`

#### 6.439.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Instead, include `<algorithm>`.

## 6.440 prefix\_search\_node\_update\_imp.hpp File Reference

#### 6.440.1 Detailed Description

Contains an implementation of `prefix_search_node_update`.

## 6.441 priority\_queue.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.441.1 Detailed Description

Contains `priority_queues`.

### 6.442 `priority_queue_base_dispatch.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binary\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, pairing\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, rc\\_binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, thin\\_heap\\_tag, null\\_type>](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`

#### 6.442.1 Detailed Description

Contains an pqiative container dispatching base.

### 6.443 `probe_fn_base.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base<\\_Alloc>](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.443.1 Detailed Description

Contains a probe policy base.

## 6.444 profiler.h File Reference

## Classes

- struct [\\_\\_gnu\\_profile::\\_\\_reentrance\\_guard](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Macros

- #define [\\_\\_profcxx\\_hash\\_func\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hash\\_func\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hashtable\\_size\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hashtable\\_size\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_hashtable\\_size\\_resize](#)(\_\_x...)
- #define [\\_\\_profcxx\\_is\\_invalid](#)()
- #define [\\_\\_profcxx\\_is\\_off](#)()
- #define [\\_\\_profcxx\\_is\\_on](#)()
- #define [\\_\\_profcxx\\_list2slist\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2slist\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2slist\\_operation](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2slist\\_rewind](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2vector\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2vector\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2vector\\_insert](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2vector\\_invalid\\_operator](#)(\_\_x...)
- #define [\\_\\_profcxx\\_list2vector\\_iterate](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_erase](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_find](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_insert](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_invalidate](#)(\_\_x...)
- #define [\\_\\_profcxx\\_map2umap\\_iterate](#)(\_\_x...)
- #define [\\_\\_profcxx\\_report](#)()
- #define [\\_\\_profcxx\\_turn\\_off](#)()
- #define [\\_\\_profcxx\\_turn\\_on](#)()
- #define [\\_\\_profcxx\\_vector2list\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector2list\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector2list\\_insert](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector2list\\_invalid\\_operator](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector2list\\_iterate](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector2list\\_resize](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector\\_size\\_construct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector\\_size\\_destruct](#)(\_\_x...)
- #define [\\_\\_profcxx\\_vector\\_size\\_resize](#)(\_\_x...)
- #define [\\_GLIBCXX\\_PROFILE\\_DATA](#)(\_\_name)

- `#define _GLIBCXX_PROFILE_DEFINE_DATA(__type, __name, __initial_value...)`
- `#define _GLIBCXX_PROFILE_DEFINE_UNINIT_DATA(__type, __name)`
- `#define _GLIBCXX_PROFILE_MAX_STACK_DEPTH`
- `#define _GLIBCXX_PROFILE_MAX_STACK_DEPTH_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT`
- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

## Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report ()`
- `__hashfunc_info * __gnu_profile::__trace_hash_func_construct ()`
- `void __gnu_profile::__trace_hash_func_destruct (__hashfunc_info *, std::size_t, std::size_t, std::size_t)`
- `__container_size_info * __gnu_profile::__trace_hashtable_size_construct (std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (__container_size_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (__container_size_info *, std::size_t, std::size_t)`
- `__list2slist_info * __gnu_profile::__trace_list_to_slist_construct ()`
- `void __gnu_profile::__trace_list_to_slist_destruct (__list2slist_info *)`
- `void __gnu_profile::__trace_list_to_slist_operation (__list2slist_info *)`
- `void __gnu_profile::__trace_list_to_slist_rewind (__list2slist_info *)`
- `__list2vector_info * __gnu_profile::__trace_list_to_vector_construct ()`
- `void __gnu_profile::__trace_list_to_vector_destruct (__list2vector_info *)`
- `void __gnu_profile::__trace_list_to_vector_insert (__list2vector_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (__list2vector_info *)`
- `void __gnu_profile::__trace_list_to_vector_iterate (__list2vector_info *, int)`
- `void __gnu_profile::__trace_list_to_vector_resize (__list2vector_info *, std::size_t, std::size_t)`
- `__map2umap_info * __gnu_profile::__trace_map_to_unordered_map_construct ()`
- `void __gnu_profile::__trace_map_to_unordered_map_destruct (__map2umap_info *)`
- `void __gnu_profile::__trace_map_to_unordered_map_erase (__map2umap_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_find (__map2umap_info *, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_insert (__map2umap_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_invalidate (__map2umap_info *)`
- `void __gnu_profile::__trace_map_to_unordered_map_iterate (__map2umap_info *, std::size_t)`
- `__container_size_info * __gnu_profile::__trace_vector_size_construct (std::size_t)`
- `void __gnu_profile::__trace_vector_size_destruct (__container_size_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_vector_size_resize (__container_size_info *, std::size_t, std::size_t)`
- `__vector2list_info * __gnu_profile::__trace_vector_to_list_construct ()`
- `void __gnu_profile::__trace_vector_to_list_destruct (__vector2list_info *)`
- `void __gnu_profile::__trace_vector_to_list_insert (__vector2list_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_vector_to_list_invalid_operator (__vector2list_info *)`
- `void __gnu_profile::__trace_vector_to_list_iterate (__vector2list_info *, int)`
- `void __gnu_profile::__trace_vector_to_list_resize (__vector2list_info *, std::size_t, std::size_t)`
- `bool __gnu_profile::__turn_off ()`
- `bool __gnu_profile::__turn_on ()`

## 6.444.1 Detailed Description

Interface of the profiling runtime library.

## 6.445 profiler\_algos.h File Reference

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Functions

- `template<typename _InputIterator, typename _Function >  
_Function __gnu_profile::__for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Container >  
void __gnu_profile::__insert_top_n (_Container &__output, const typename _Container::value_type &__value,  
typename _Container::size_type __n)`
- `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator __gnu_profile::__remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵  
value)`
- `template<typename _Container >  
void __gnu_profile::__top_n (const _Container &__input, _Container &__output, typename _Container::size_↵  
type __n)`

## 6.445.1 Detailed Description

Algorithms used by the profile extension.

This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

## 6.446 profiler\_container\_size.h File Reference

## Classes

- class [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_stack\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_container\\_size](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)



#### 6.446.1 Detailed Description

Diagnostics for container sizes.

### 6.447 profiler\_hash\_func.h File Reference

#### Classes

- class [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_stack\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func](#)

#### Namespaces

- [\\_\\_gnu\\_profile](#)

#### Functions

- `__hashfunc_info * __gnu_profile::__trace_hash_func_construct ()`
- `void __gnu_profile::__trace_hash_func_destruct (__hashfunc_info *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hash_func_free ()`
- `void __gnu_profile::__trace_hash_func_init ()`
- `void __gnu_profile::__trace_hash_func_report (FILE * __f, __warning_vector_t & __warnings)`

#### 6.447.1 Detailed Description

Data structures to represent profiling traces.

### 6.448 profiler\_hashtable\_size.h File Reference

#### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size](#)

#### Namespaces

- [\\_\\_gnu\\_profile](#)

#### Functions

- `__container_size_info * __gnu_profile::__trace_hashtable_size_construct (std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (__container_size_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_free ()`
- `void __gnu_profile::__trace_hashtable_size_init ()`
- `void __gnu_profile::__trace_hashtable_size_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_hashtable_size_resize (__container_size_info *, std::size_t, std::size_t)`

## 6.448.1 Detailed Description

Collection of hashtable size traces.

## 6.449 profiler\_list\_to\_slist.h File Reference

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Functions

- `__list2slist_info * __gnu_profile::__trace_list_to_slist_construct ()`
- `void __gnu_profile::__trace_list_to_slist_destruct (__list2slist_info *)`
- `void __gnu_profile::__trace_list_to_slist_free ()`
- `void __gnu_profile::__trace_list_to_slist_init ()`
- `void __gnu_profile::__trace_list_to_slist_operation (__list2slist_info *)`
- `void __gnu_profile::__trace_list_to_slist_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_slist_rewind (__list2slist_info *)`

## 6.449.1 Detailed Description

Diagnostics for list to slist.

## 6.450 profiler\_list\_to\_vector.h File Reference

## Classes

- class [\\_\\_gnu\\_profile::\\_\\_list2vector\\_info](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Functions

- `__list2vector_info * __gnu_profile::__trace_list_to_vector_construct ()`
- `void __gnu_profile::__trace_list_to_vector_destruct (__list2vector_info *)`
- `void __gnu_profile::__trace_list_to_vector_free ()`
- `void __gnu_profile::__trace_list_to_vector_init ()`
- `void __gnu_profile::__trace_list_to_vector_insert (__list2vector_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (__list2vector_info *)`
- `void __gnu_profile::__trace_list_to_vector_iterate (__list2vector_info *, int)`
- `void __gnu_profile::__trace_list_to_vector_report (FILE * __f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_vector_resize (__list2vector_info *, std::size_t, std::size_t)`

#### 6.450.1 Detailed Description

diagnostics for list to vector.

### 6.451 profiler\_map\_to\_unordered\_map.h File Reference

#### Classes

- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_stack\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_map2umap](#)

#### Namespaces

- [\\_\\_gnu\\_profile](#)

#### Functions

- int [\\_\\_gnu\\_profile::\\_\\_log2](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_erase\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_find\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_insert\\_cost](#) (std::size\_t \_\_size)
- [\\_\\_map2umap\\_info \\*](#) [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_construct](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_destruct](#) ([\\_\\_map2umap\\_info \\*](#))
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_erase](#) ([\\_\\_map2umap\\_info \\*](#), std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_find](#) ([\\_\\_map2umap\\_info \\*](#), std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_free](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_insert](#) ([\\_\\_map2umap\\_info \\*](#), std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_invalidate](#) ([\\_\\_map2umap\\_info \\*](#))
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_iterate](#) ([\\_\\_map2umap\\_info \\*](#) [\\_\\_info](#), int)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_report](#) (FILE \* [\\_\\_f](#), [\\_\\_warning\\_vector\\_t](#) & [\\_\\_warnings](#))

#### 6.451.1 Detailed Description

Diagnostics for map to unordered\_map.

### 6.452 profiler\_node.h File Reference

#### Classes

- class [\\_\\_gnu\\_profile::\\_\\_object\\_info\\_base](#)
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_hash](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Typedefs

- typedef void \* **\_\_gnu\_profile::\_\_instruction\_address\_t**
- typedef std::vector< \_\_instruction\_address\_t > **\_\_gnu\_profile::\_\_stack\_npt**
- typedef \_\_stack\_npt \* **\_\_gnu\_profile::\_\_stack\_t**

## Functions

- **\_\_stack\_t \_\_gnu\_profile::\_\_get\_stack ()**
- **std::size\_t \_\_gnu\_profile::\_\_size (\_\_stack\_t \_\_stack)**
- **std::size\_t \_\_gnu\_profile::\_\_stack\_max\_depth ()**
- **void \_\_gnu\_profile::\_\_write (FILE \* \_\_f, \_\_stack\_t \_\_stack)**

## 6.452.1 Detailed Description

Data structures to represent a single profiling event.

## 6.453 profiler\_state.h File Reference

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Enumerations

- enum **\_\_state\_type** { **\_\_ON**, **\_\_OFF**, **\_\_INVALID** }

## Functions

- **bool \_\_gnu\_profile::\_\_is\_invalid ()**
- **bool \_\_gnu\_profile::\_\_is\_off ()**
- **bool \_\_gnu\_profile::\_\_is\_on ()**
- **bool \_\_gnu\_profile::\_\_turn (\_\_state\_type \_\_s)**
- **bool \_\_gnu\_profile::\_\_turn\_off ()**
- **bool \_\_gnu\_profile::\_\_turn\_on ()**
- **\_\_gnu\_profile::\_\_GLIBCXX\_PROFILE\_DEFINE\_DATA (\_\_state\_type, \_\_state, \_\_INVALID)**

## 6.453.1 Detailed Description

Global profiler state.

## 6.454 profiler\_trace.h File Reference

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_base< \\_\\_object\\_info, \\_\\_stack\\_info >](#)
- struct [\\_\\_gnu\\_profile::\\_\\_warning\\_data](#)

### Namespaces

- [\\_\\_gnu\\_profile](#)

### Macros

- `#define \_GLIBCXX\_IMPL\_UNORDERED\_MAP`

### Typedefs

- typedef std::vector< \_\_cost\_factor \* > [\\_\\_gnu\\_profile::\\_\\_cost\\_factor\\_vector](#)
- typedef std::unordered\_map< [std::string](#), [std::string](#) > [\\_\\_gnu\\_profile::\\_\\_env\\_t](#)
- typedef std::vector< \_\_warning\_data > [\\_\\_gnu\\_profile::\\_\\_warning\\_vector\\_t](#)

### Functions

- std::size\_t [\\_\\_gnu\\_profile::\\_\\_env\\_to\\_size\\_t](#) (const char \* \_\_env\_var, std::size\_t \_\_default\_value)
- int [\\_\\_gnu\\_profile::\\_\\_log\\_magnitude](#) (float \_\_f)
- std::size\_t [\\_\\_gnu\\_profile::\\_\\_max\\_mem](#) ()
- FILE \* [\\_\\_gnu\\_profile::\\_\\_open\\_output\\_file](#) (const char \* \_\_extension)
- bool [\\_\\_gnu\\_profile::\\_\\_profcxx\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_profcxx\\_init\\_unconditional](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_read\\_cost\\_factors](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_report](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_report\\_and\\_free](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_set\\_cost\\_factors](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_set\\_max\\_mem](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_set\\_max\\_stack\\_trace\\_depth](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_set\\_max\\_warn\\_count](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_set\\_trace\\_path](#) ()
- std::size\_t [\\_\\_gnu\\_profile::\\_\\_stack\\_max\\_depth](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_free](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_report](#) (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_free](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_report](#) (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_free](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_slist\\_report](#) (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)

- void `__gnu_profile::__trace_list_to_vector_free()`
- void `__gnu_profile::__trace_list_to_vector_init()`
- void `__gnu_profile::__trace_list_to_vector_report(FILE * __f, __warning_vector_t & __warnings)`
- void `__gnu_profile::__trace_map_to_unordered_map_free()`
- void `__gnu_profile::__trace_map_to_unordered_map_init()`
- void `__gnu_profile::__trace_map_to_unordered_map_report(FILE * __f, __warning_vector_t & __warnings)`
- `template<typename __object_info, typename __stack_info >`  
void `__gnu_profile::__trace_report(__trace_base< __object_info, __stack_info > * __cont, FILE * __f, __warning_vector_t & __warnings)`
- void `__gnu_profile::__trace_vector_size_free()`
- void `__gnu_profile::__trace_vector_size_init()`
- void `__gnu_profile::__trace_vector_size_report(FILE *, __warning_vector_t &)`
- void `__gnu_profile::__trace_vector_to_list_free()`
- void `__gnu_profile::__trace_vector_to_list_init()`
- void `__gnu_profile::__trace_vector_to_list_report(FILE *, __warning_vector_t &)`
- void `__gnu_profile::__write_cost_factors()`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_hash_func *, __S_hash_func, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_hashtable_size *, __S_hashtable_size, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_map2umap *, __S_map2umap, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_vector_size *, __S_vector_size, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_vector_to_list *, __S_vector_to_list, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_list_to_slist *, __S_list_to_slist, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__trace_list_to_vector *, __S_list_to_vector, 0)`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __vector_shift_cost_factor, {"__vector_↵_shift_cost_factor", 1.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __vector_iterate_cost_factor, {"__↵vector_iterate_cost_factor", 1.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __vector_resize_cost_factor, {"__↵vector_resize_cost_factor", 1.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __list_shift_cost_factor, {"__list_shift_↵cost_factor", 0.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __list_iterate_cost_factor, {"__list_↵iterate_cost_factor", 10.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __list_resize_cost_factor, {"__list_↵resize_cost_factor", 0.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __map_insert_cost_factor, {"__map_↵insert_cost_factor", 1.5})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __map_erase_cost_factor, {"__map_↵erase_cost_factor", 1.5})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __map_find_cost_factor, {"__map_find_↵cost_factor", 1})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __map_iterate_cost_factor, {"__map_↵iterate_cost_factor", 2.3})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __umap_insert_cost_factor, {"__umap_↵insert_cost_factor", 12.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __umap_erase_cost_factor, {"__umap_↵erase_cost_factor", 12.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __umap_find_cost_factor, {"__umap_↵find_cost_factor", 10.0})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor, __umap_iterate_cost_factor, {"__↵umap_iterate_cost_factor", 1.7})`
- `__gnu_profile::GLIBCXX_PROFILE_DEFINE_DATA(__cost_factor_vector *, __cost_factors, 0)`

- [\\_\\_gnu\\_profile::\\_\\_GLIBCXX\\_PROFILE\\_DEFINE\\_DATA](#) (const char \*, \_S\_trace\_file\_name, \_GLIBCXX\_PROFILE\_TRACE\_PATH\_ROOT)
- [\\_\\_gnu\\_profile::\\_\\_GLIBCXX\\_PROFILE\\_DEFINE\\_DATA](#) (std::size\_t, \_S\_max\_warn\_count, \_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT)
- [\\_\\_gnu\\_profile::\\_\\_GLIBCXX\\_PROFILE\\_DEFINE\\_DATA](#) (std::size\_t, \_S\_max\_stack\_depth, \_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH)
- [\\_\\_gnu\\_profile::\\_\\_GLIBCXX\\_PROFILE\\_DEFINE\\_DATA](#) (std::size\_t, \_S\_max\_mem, \_GLIBCXX\_PROFILE\_MEMORY\_PER\_DIAGNOSTIC)
- [\\_\\_gnu\\_profile::\\_\\_GLIBCXX\\_PROFILE\\_DEFINE\\_UNINIT\\_DATA](#) (\_\_env\_t, \_\_env)
- [\\_\\_gnu\\_profile::\\_\\_GLIBCXX\\_PROFILE\\_DEFINE\\_UNINIT\\_DATA](#) (\_\_gnu\_cxx::\_\_mutex, \_\_global\_mutex)

#### 6.454.1 Detailed Description

Data structures to represent profiling traces.

### 6.455 profiler\_vector\_size.h File Reference

#### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size](#)

#### Namespaces

- [\\_\\_gnu\\_profile](#)

#### Functions

- [\\_\\_container\\_size\\_info \\* \\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_construct](#) (std::size\_t)
- [void \\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_destruct](#) (\_\_container\_size\_info \*, std::size\_t, std::size\_t)
- [void \\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_free](#) ()
- [void \\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_init](#) ()
- [void \\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_report](#) (FILE \*, \_\_warning\_vector\_t &)
- [void \\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_resize](#) (\_\_container\_size\_info \*, std::size\_t, std::size\_t)

#### 6.455.1 Detailed Description

Collection of vector size traces.

### 6.456 profiler\_vector\_to\_list.h File Reference

#### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list](#)
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_info](#)
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_stack\\_info](#)

## Namespaces

- [\\_\\_gnu\\_profile](#)

## Functions

- `__vector2list_info * __gnu_profile::__trace_vector_to_list_construct ()`
- `void __gnu_profile::__trace_vector_to_list_destruct (__vector2list_info *)`
- `void __gnu_profile::__trace_vector_to_list_free ()`
- `void __gnu_profile::__trace_vector_to_list_init ()`
- `void __gnu_profile::__trace_vector_to_list_insert (__vector2list_info *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_vector_to_list_invalid_operator (__vector2list_info *)`
- `void __gnu_profile::__trace_vector_to_list_iterate (__vector2list_info *, int)`
- `void __gnu_profile::__trace_vector_to_list_report (FILE *, __warning_vector_t &)`
- `void __gnu_profile::__trace_vector_to_list_resize (__vector2list_info *, std::size_t, std::size_t)`

### 6.456.1 Detailed Description

diagnostics for vector to list.

## 6.457 propagate\_const File Reference

## Classes

- class [std::experimental::fundamentals\\_v2::propagate\\_const<\\_Tp>](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_PROPAGATE_CONST`



## Functions

- `template<typename _Tp >`  
`constexpr const _Tp & std::experimental::fundamentals_v2::get_underlying (const propagate_const< _Tp >`  
`&__pt) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp & std::experimental::fundamentals_v2::get_underlying (propagate_const< _Tp > &__pt)`  
`noexcept`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt,`  
`nullptr_t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (nullptr_t, const propagate_const< _Tp >`  
`&__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, const`  
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator!= (const _Tp &__t, const propagate_const< ↵`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const`  
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const _Tp &__t, const propagate_const< ↵`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate_const< _Tp > &__pt,`  
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const _Tp &__t, const propagate_const<`  
`_Up > &__pu)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`  
`nullptr_t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (nullptr_t, const propagate_const< _Tp >`  
`&__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`  
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`  
`const _Up &__u)`

- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const _Tp &__t, const propagate_const<_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate_const<_Tp > &__pt, const propagate_const<_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate_const<_Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const _Tp &__t, const propagate_const<_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate_const<_Tp > &__pt, const propagate_const<_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate_const<_Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const _Tp &__t, const propagate_const<_Up > &__pu)`
- `template<typename _Tp >`  
`constexpr void std::experimental::fundamentals_v2::swap (propagate_const<_Tp > &__pt, propagate_const<_Tp > &__pt2) noexcept(__is_nothrow_swappable<_Tp >::value)`

#### 6.457.1 Detailed Description

This is a TS C++ Library header.

## 6.458 ptr\_traits.h File Reference

### Classes

- struct [std::pointer\\_traits<\\_Ptr >](#)
- struct [std::pointer\\_traits<\\_Tp \\* >](#)

### Namespaces

- [std](#)

### Typedefs

- `template<typename _Tp >`  
`using std::__get_first_arg_t = typename __get_first_arg<_Tp >::type`
- `template<typename _Tp >`  
`using std::__make_not_void = typename conditional< is_void<_Tp >::value, __undefined, _Tp >::type`
- `template<typename _Ptr, typename _Tp >`  
`using std::__ptr_rebind = typename pointer_traits<_Ptr >::template rebind<_Tp >`
- `template<typename _Tp, typename _Up >`  
`using std::__replace_first_arg_t = typename __replace_first_arg<_Tp, _Up >::type`

## Functions

- `template<typename _Tp >`  
`constexpr _Tp * std::__to_address (_Tp * __ptr) noexcept`
- `template<typename _Ptr >`  
`constexpr std::pointer\_traits< _Ptr >::element_type * std::__to_address (const _Ptr & __ptr)`

### 6.458.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.459 `quadratic_probe_fn_imp.hpp` File Reference

### 6.459.1 Detailed Description

Contains a probe policy implementation

## 6.460 `queue` File Reference

## Macros

- `#define _GLIBCXX_QUEUE`

### 6.460.1 Detailed Description

This is a Standard C++ Library header.

## 6.461 `queue.h` File Reference

## Classes

- class [\\_\\_gnu\\_parallel::\\_\\_RestrictedBoundedConcurrentQueue](#)< \_Tp >

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_VOLATILE`

### 6.461.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### 6.461.2 Macro Definition Documentation

#### 6.461.2.1 \_GLIBCXX\_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file queue.h.

## 6.462 quicksort.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __↵  
__num_threads)`
- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, __↵  
_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >  
std::iterator_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_divide (_RAIter __begin, __↵  
_RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, type-  
name std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

### 6.462.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

## 6.463 quoted\_string.h File Reference

### Classes

- struct `std::__detail::Quoted_string<_String, _CharT >`

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<typename _CharT, typename _Traits >  
std::basic\_ostream< _CharT, _Traits > & std::\_\_detail::operator<< (std::basic\_ostream< _CharT, _Traits > &↔  
__os, const _Quoted_string< const _CharT *, _CharT > &__str)`
- `template<typename _CharT, typename _Traits, typename _String >  
std::basic\_ostream< _CharT, _Traits > & std::\_\_detail::operator<< (std::basic\_ostream< _CharT, _Traits > &↔  
__os, const _Quoted_string< _String, _CharT > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
std::basic\_istream< _CharT, _Traits > & std::\_\_detail::operator>> (std::basic\_istream< _CharT, _Traits > &↔  
__is, const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)`

### 6.463.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iomanip>`.

## 6.464 `r_erase_fn_imps.hpp` File Reference

### 6.464.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.465 `r_erase_fn_imps.hpp` File Reference

### 6.465.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.466 `random` File Reference

## Macros

- `#define _GLIBCXX_RANDOM`

### 6.466.1 Detailed Description

This is a Standard C++ Library header.

## 6.467 random File Reference

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_experimental_randint`
- `#define _GLIBCXX_EXPERIMENTAL_RANDOM`

### Functions

- [std::default\\_random\\_engine](#) & [std::experimental::fundamentals\\_v2::S\\_randint\\_engine](#) ()
- `template<typename _IntType >`  
`_IntType std::experimental::fundamentals\_v2::randint (_IntType __a, _IntType __b)`
- `void std::experimental::fundamentals\_v2::reseed ()`
- `void std::experimental::fundamentals\_v2::reseed (default_random_engine::result_type __value)`

#### 6.467.1 Detailed Description

This is a TS C++ Library header.

## 6.468 random.h File Reference

### Classes

- class [std::bernoulli\\_distribution](#)
- struct [std::bernoulli\\_distribution::param\\_type](#)
- class [std::binomial\\_distribution< \\_IntType >](#)
- struct [std::binomial\\_distribution< \\_IntType >::param\\_type](#)
- class [std::cauchy\\_distribution< \\_RealType >](#)
- struct [std::cauchy\\_distribution< \\_RealType >::param\\_type](#)
- class [std::chi\\_squared\\_distribution< \\_RealType >](#)
- struct [std::chi\\_squared\\_distribution< \\_RealType >::param\\_type](#)
- class [std::discard\\_block\\_engine< \\_RandomNumberEngine, \\_\\_p, \\_\\_r >](#)
- class [std::discrete\\_distribution< \\_IntType >](#)
- struct [std::discrete\\_distribution< \\_IntType >::param\\_type](#)
- class [std::exponential\\_distribution< \\_RealType >](#)
- struct [std::exponential\\_distribution< \\_RealType >::param\\_type](#)
- class [std::extreme\\_value\\_distribution< \\_RealType >](#)
- struct [std::extreme\\_value\\_distribution< \\_RealType >::param\\_type](#)
- class [std::fisher\\_f\\_distribution< \\_RealType >](#)
- struct [std::fisher\\_f\\_distribution< \\_RealType >::param\\_type](#)
- class [std::gamma\\_distribution< \\_RealType >](#)
- struct [std::gamma\\_distribution< \\_RealType >::param\\_type](#)

- class [std::geometric\\_distribution< \\_IntType >](#)
- struct [std::geometric\\_distribution< \\_IntType >::param\\_type](#)
- class [std::independent\\_bits\\_engine< \\_RandomNumberEngine, \\_\\_w, \\_UIntType >](#)
- class [std::linear\\_congruential\\_engine< \\_UIntType, \\_\\_a, \\_\\_c, \\_\\_m >](#)
- class [std::lognormal\\_distribution< \\_RealType >](#)
- struct [std::lognormal\\_distribution< \\_RealType >::param\\_type](#)
- class [std::mersenne\\_twister\\_engine< \\_UIntType, \\_\\_w, \\_\\_n, \\_\\_m, \\_\\_r, \\_\\_a, \\_\\_u, \\_\\_d, \\_\\_s, \\_\\_b, \\_\\_t, \\_\\_c, \\_\\_l, \\_\\_f >](#)
- class [std::negative\\_binomial\\_distribution< \\_IntType >](#)
- struct [std::negative\\_binomial\\_distribution< \\_IntType >::param\\_type](#)
- class [std::normal\\_distribution< \\_RealType >](#)
- struct [std::normal\\_distribution< \\_RealType >::param\\_type](#)
- class [std::piecewise\\_constant\\_distribution< \\_RealType >](#)
- struct [std::piecewise\\_constant\\_distribution< \\_RealType >::param\\_type](#)
- class [std::piecewise\\_linear\\_distribution< \\_RealType >](#)
- struct [std::piecewise\\_linear\\_distribution< \\_RealType >::param\\_type](#)
- class [std::poisson\\_distribution< \\_IntType >](#)
- struct [std::poisson\\_distribution< \\_IntType >::param\\_type](#)
- class [std::random\\_device](#)
- class [std::seed\\_seq](#)
- class [std::shuffle\\_order\\_engine< \\_RandomNumberEngine, \\_\\_k >](#)
- class [std::student\\_t\\_distribution< \\_RealType >](#)
- struct [std::student\\_t\\_distribution< \\_RealType >::param\\_type](#)
- class [std::subtract\\_with\\_carry\\_engine< \\_UIntType, \\_\\_w, \\_\\_s, \\_\\_r >](#)
- class [std::uniform\\_real\\_distribution< \\_RealType >](#)
- struct [std::uniform\\_real\\_distribution< \\_RealType >::param\\_type](#)
- class [std::weibull\\_distribution< \\_RealType >](#)
- struct [std::weibull\\_distribution< \\_RealType >::param\\_type](#)

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Typedefs

- typedef minstd\_rand0 **std::default\_random\_engine**
- typedef shuffle\_order\_engine< minstd\_rand0, 256 > **std::knuth\_b**
- typedef linear\_congruential\_engine< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > [std::minstd\\_rand](#)
- typedef linear\_congruential\_engine< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [std::minstd\\_rand0](#)
- typedef mersenne\_twister\_engine< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef mersenne\_twister\_engine< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > [std::mt19937\\_64](#)
- typedef discard\_block\_engine< ranlux24\_base, 223, 23 > **std::ranlux24**
- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > **std::ranlux24\_base**
- typedef discard\_block\_engine< ranlux48\_base, 389, 11 > **std::ranlux48**
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > **std::ranlux48\_base**

## Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >  
_RealType std::generate\_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>  
bool std::operator!= (const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lhs, const  
std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _U↵  
IntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>  
bool std::operator!= (const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s,  
__b, __t, __c, __l, __f > &__lhs, const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u,  
__d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>  
bool std::operator!= (const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__lhs, const  
std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
bool std::operator!= (const std::discard\_block\_engine< _RandomNumberEngine, __p, __r > &__lhs, const  
std::discard\_block\_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >  
bool std::operator!= (const std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType > &__lhs,  
const std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>  
bool std::operator!= (const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__lhs, const  
std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _IntType >  
bool std::operator!= (const std::uniform\_int\_distribution< _IntType > &__d1, const std::uniform\_int\_distribution<  
_IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator!= (const std::uniform\_real\_distribution< _IntType > &__d1, const std::uniform\_real\_distribution<  
_IntType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::normal\_distribution< _RealType > &__d1, const std::normal\_distribution< _↵  
RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::lognormal\_distribution< _RealType > &__d1, const std::lognormal\_distribution<  
_RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::gamma\_distribution< _RealType > &__d1, const std::gamma\_distribution< _↵  
RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::chi\_squared\_distribution< _RealType > &__d1, const std::chi\_squared\_distribution<  
_RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::cauchy\_distribution< _RealType > &__d1, const std::cauchy\_distribution< _↵  
RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::fisher\_f\_distribution< _RealType > &__d1, const std::fisher\_f\_distribution< _↵  
RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::student\_t\_distribution< _RealType > &__d1, const std::student\_t\_distribution<  
_RealType > &__d2)`
- `bool std::operator!= (const std::bernoulli\_distribution &__d1, const std::bernoulli\_distribution &__d2)`



- `template<typename _IntType >`  
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &  
std::uniform_int_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &  
std::uniform_real_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::extreme_value_distribution< _RealType > &__x)`

### 6.468.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.469 random.tcc File Reference

### Namespaces

- `std`
- `std::__detail`

### Macros

- `#define _RANDOM_TCC`

## Functions

- `template<typename _ValT, typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::detail::extract_params (basic_istream< _CharT, _Traits > & __is,`  
`vector< _ValT > & __vals, size_t __n)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::detail::normalize (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`const _Tp & __factor)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`  
`_RealType std::generate_canonical (_UniformRandomNumberGenerator & __g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const linear_congruential_engine< _UIntType, __a, __c, __m > & __lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _U←`  
`IntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f`  
`> & __x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const discard_block_engine< _RandomNumberEngine, __p, __r > & __x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const shuffle_order_engine< _RandomNumberEngine, __k > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const negative_binomial_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const poisson_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const binomial_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const normal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const lognormal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const chi_squared_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const fisher_f_distribution< _RealType > & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const gamma_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType >`  
`bool std::operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _↵`  
`RealType > &__d2)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__↵`  
`is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _U↵`  
`IntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__↵`  
`is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`  
`&__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__↵`  
`is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__↵`  
`is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__↵`  
`is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)`



- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::extreme_value_distribution< _RealType > &__x)`

### 6.469.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.470 random.tcc File Reference

### Namespaces

- [`\_\_gnu\_cxx`](#)

### Macros

- `#define _EXT_RANDOM_TCC`

### Functions

- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const logistic_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &←`  
`__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`  
`bool gnu_cxx::operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __←`  
`pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4`  
`> &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __←`  
`sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<size_t _Dimen, typename _RealType >`  
`bool gnu_cxx::operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const`  
`__gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT`  
`, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2,`  
`__msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, rice_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, k_distribution< _RealType > &__x)`



- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, logistic_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &←`  
`__is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`

#### 6.470.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/random>`.

## 6.471 random\_number.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RandomNumber](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### 6.471.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.



## 6.472 random\_shuffle.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter >](#)
- struct [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

### Functions

- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator \_\_rng=\_RandomNumber())
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs](#) (\_RAIter \_\_begin, \_RAIter \_\_end, typename std::iterator\_traits<\_RAIter>::difference\_type \_\_n, \_ThreadIndex \_\_num\_threads, \_RandomNumberGenerator &\_\_rng)
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu](#) (\_DRSSorterPU<\_RAIter, \_RandomNumberGenerator > \* \_\_pus)
- template<typename \_RandomNumberGenerator >  
int [\\_\\_gnu\\_parallel::\\_\\_random\\_number\\_pow2](#) (int \_\_logp, \_RandomNumberGenerator &\_\_rng)
- template<typename \_Tp >  
[\\_Tp \\_\\_gnu\\_parallel::\\_\\_round\\_up\\_to\\_pow2](#) (\_Tp \_\_x)
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rng)

### 6.472.1 Detailed Description

Parallel implementation of std::random\_shuffle(). This file is a GNU parallel extension to the Standard C++ Library.

## 6.473 range\_access.h File Reference

### Classes

- class [std::valarray<\\_Tp >](#)

## Namespaces

- [std](#)

## Functions

- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Tp, size_t _Nm>  
_GLIBCXX14_CONSTEXPR _Tp * std::begin (_Tp(&__arr)[_Nm])`
- `template<class _Tp >  
_Tp * std::begin (valarray< _Tp > &__va)`
- `template<class _Tp >  
const _Tp * std::begin (const valarray< _Tp > &__va)`
- `template<typename _Container >  
constexpr auto std::cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(__cont))`
- `template<typename _Container >  
constexpr auto std::cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(__cont))`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::crbegin (const _Container &__cont) -> decltype(std::rbegin(__cont))`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::crend (const _Container &__cont) -> decltype(std::rend(__cont))`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::end (_Container &__cont) -> decltype(__cont.end())`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::end (const _Container &__cont) -> decltype(__cont.end())`
- `template<typename _Tp, size_t _Nm>  
_GLIBCXX14_CONSTEXPR _Tp * std::end (_Tp(&__arr)[_Nm])`
- `template<class _Tp >  
_Tp * std::end (valarray< _Tp > &__va)`
- `template<class _Tp >  
const _Tp * std::end (const valarray< _Tp > &__va)`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::rbegin (_Container &__cont) -> decltype(__cont.rbegin())`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::rbegin (const _Container &__cont) -> decltype(__cont.rbegin())`
- `template<typename _Tp, size_t _Nm>  
_GLIBCXX17_CONSTEXPR reverse_iterator< _Tp * > std::rbegin (_Tp(&__arr)[_Nm])`
- `template<typename _Tp >  
_GLIBCXX17_CONSTEXPR reverse_iterator< const _Tp * > std::rbegin (initializer_list< _Tp > __il)`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::rend (_Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Container >  
_GLIBCXX17_CONSTEXPR auto std::rend (const _Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>  
_GLIBCXX17_CONSTEXPR reverse_iterator< _Tp * > std::rend (_Tp(&__arr)[_Nm])`
- `template<typename _Tp >  
_GLIBCXX17_CONSTEXPR reverse_iterator< const _Tp * > std::rend (initializer_list< _Tp > __il)`

#### 6.473.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

#### 6.474 `ranged_hash_fn.hpp` File Reference

##### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, Store\\_Hash >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)

##### Namespaces

- [\\_\\_gnu\\_pbds](#)

##### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.474.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

#### 6.475 `ranged_probe_fn.hpp` File Reference

##### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, Store\\_Hash >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 6.475.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probing.

## 6.476 ratio File Reference

## Classes

- struct [std::ratio<\\_Num, \\_Den>](#)
- struct [std::ratio\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_not\\_equal<\\_R1, \\_R2>](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_RATIO`

## Typedefs

- `template<typename _R1, typename _R2>`  
  using [std::ratio\\_divide](#) = typename `__ratio_divide<_R1, _R2>::type`
- `template<typename _R1, typename _R2>`  
  using [std::ratio\\_multiply](#) = typename `__ratio_multiply<_R1, _R2>::type`

### 6.476.1 Detailed Description

This is a Standard C++ Library header.

## 6.477 ratio File Reference

### Namespaces

- [std](#)
- [std::tr2](#)

### 6.477.1 Detailed Description

This is a TR2 C++ Library header.

## 6.478 ratio File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_RATIO`

### Variables

- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_equal_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_greater_equal_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_greater_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_less_equal_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_less_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_not_equal_v`

### 6.478.1 Detailed Description

This is a TS C++ Library header.

## 6.479 rb\_tree File Reference

### Classes

- `struct \_\_gnu\_cxx::rb\_tree< \_Key, \_Value, \_KeyOfValue, \_Compare, \_Alloc >`

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _RB_TREE`

### 6.479.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.480 rb\_tree.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

### 6.480.1 Detailed Description

Contains an implementation for Red Black trees.

## 6.481 rc.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::rc](#)< \_Node, \_Alloc >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.481.1 Detailed Description

Contains a redundant (binary counter).

## 6.482 rc\_binomial\_heap.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RC_C_DEC`

### 6.482.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

## 6.483 rc\_string\_base.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base< \\_CharT, \\_Traits, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

### 6.483.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.484 refwrap.h File Reference

## Classes

- struct [std::\\_Maybe\\_get\\_result\\_type<\\_Functor, typename >](#)
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_ArgTypes >](#)
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_T1 >](#)
- struct [std::\\_Maybe\\_unary\\_or\\_binary\\_function<\\_Res, \\_T1, \\_T2 >](#)
- struct [std::\\_Reference\\_wrapper\\_base<\\_Tp >](#)
- struct [std::\\_Weak\\_result\\_type<\\_Functor >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl<\\_Functor >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl<\\_Res\(\\*\)\(\\_ArgTypes...\) \\_GLIBCXX\\_NOEXCEPT\\_QUAL >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl<\\_Res\(\\*\)\(\\_ArgTypes.....\) \\_GLIBCXX\\_NOEXCEPT\\_QUAL >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl<\\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_NOEXCEPT\\_QUAL >](#)
- struct [std::\\_Weak\\_result\\_type\\_impl<\\_Res\(\\_ArgTypes.....\) \\_GLIBCXX\\_NOEXCEPT\\_QUAL >](#)
- class [std::reference\\_wrapper<\\_Tp >](#)

## Namespaces

- [std](#)

## Macros

- `#define \_GLIBCXX\_MEM\_FN\_TRAITS(_REF, _LVAL, _RVAL)`
- `#define \_GLIBCXX\_MEM\_FN\_TRAITS2(_CV, _REF, _LVAL, _RVAL)`

## Functions

- `template<typename _Tp >`  
`reference_wrapper<_Tp > std::ref (_Tp &__t) noexcept`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (const _Tp &__t) noexcept`
- `template<typename _Tp >`  
`void std::ref (const _Tp &&)=delete`
- `template<typename _Tp >`  
`void std::cref (const _Tp &&)=delete`
- `template<typename _Tp >`  
`reference_wrapper<_Tp > std::ref (reference_wrapper<_Tp > __t) noexcept`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (reference_wrapper<_Tp > __t) noexcept`

## 6.484.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.



## 6.485 regex File Reference

### Macros

- `#define _GLIBCXX_REGEX`

### 6.485.1 Detailed Description

This is a Standard C++ Library header.

## 6.486 regex File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_REGEX`

### 6.486.1 Detailed Description

This is a TS C++ Library header.

## 6.487 regex.h File Reference

### Classes

- class [std::\\_\\_detail::\\_Executor<\\_Bilter, \\_Alloc, \\_TraitsT, \\_\\_dfs\\_mode >](#)
- class [std::basic\\_regex<\\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::basic\\_regex<\\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::match\\_results<\\_Bi\\_iter, \\_Alloc >](#)
- class [std::match\\_results<\\_Bi\\_iter, \\_Alloc >](#)
- class [std::regex\\_iterator<\\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::regex\\_token\\_iterator<\\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >](#)
- class [std::regex\\_traits<\\_Ch\\_type >](#)
- class [std::sub\\_match<\\_Bilter >](#)

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Typedefs

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`using std::__sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits,`  
`_Ch_alloc >`
- `typedef match_results< const char * > std::cmatch`
- `typedef regex_iterator< const char * > std::cregex_iterator`
- `typedef regex_token_iterator< const char * > std::cregex_token_iterator`
- `typedef sub_match< const char * > std::csub_match`
- `typedef basic_regex< char > std::regex`
- `typedef match_results< string::const_iterator > std::smatch`
- `typedef regex_iterator< string::const_iterator > std::sregex_iterator`
- `typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator`
- `typedef sub_match< string::const_iterator > std::ssub_match`
- `typedef match_results< const wchar_t * > std::wcmatch`
- `typedef regex_iterator< const wchar_t * > std::wcregex_iterator`
- `typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator`
- `typedef sub_match< const wchar_t * > std::wcsb_match`
- `typedef basic_regex< wchar_t > std::wregex`
- `typedef match_results< wstring::const_iterator > std::wsmatch`
- `typedef regex_iterator< wstring::const_iterator > std::wsregex_iterator`
- `typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator`
- `typedef sub_match< wstring::const_iterator > std::wssub_match`

## Enumerations

- `enum _RegexExecutorPolicy : int { _S_auto, _S_alternate }`

## Functions

- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT, _RegexExecutorPolicy __policy, bool __match_↔`  
`mode>`  
`bool std::__detail::__regex_algo_impl (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > &__m, const`  
`basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags)`
- `template<typename _Bilter >`  
`bool std::operator!= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`bool std::operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`  
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _↔`  
`Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter`  
`> &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`  
`const * __rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter`  
`> &__rhs)`

- `template<typename _Bi_iter >`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`  
`const &__rhs)`
- `template<typename _Bi_iter, class _Alloc >`  
`bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc >`  
`&__m2)`
- `template<typename _Bilter >`  
`bool std::operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`  
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _`  
`Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter`  
`> &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`  
`const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter`  
`> &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`  
`const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`  
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os,`  
`const sub_match< _Bi_iter > &__m)`
- `template<typename _Bilter >`  
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`  
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits,`  
`_Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter`  
`> &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`  
`const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter`  
`> &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`  
`const &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`  
`_Bi_iter > &__rhs)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`  
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bilter >`  
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`

- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`  
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match ( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match ( _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`  
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search ( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search ( _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`  
`bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`

- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-`  
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<`  
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< type-`  
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _`  
`Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`_Out_iter std::regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`  
`_Out_iter std::regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::`  
`match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`  
`basic_string< _Ch_type, _St, _Sa > std::regex\_replace (const basic_string< _Ch_type, _St, _Sa > &__s,`  
`const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type, _St, _Sa > std::regex\_replace (const basic_string< _Ch_type, _St, _Sa > &__s,`  
`const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_`  
`type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type > std::regex\_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_`  
`traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`  
`basic_string< _Ch_type > std::regex\_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_`  
`traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_`  
`default)`

### 6.487.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.488 regex.tcc File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT, _RegexExecutorPolicy __policy, bool __match_↵ mode>`  
`bool std::__detail::__regex_algo_impl (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > &__m, const basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags)`

## Matching, Searching, and Replacing

- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::↵ ::match_default)`

### 6.488.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 6.489 regex\_automaton.h File Reference

#### Classes

- class `std::__detail::_StateSeq< _TraitsT >`

#### Namespaces

- `std`
- `std::__detail`

#### Macros

- `#define _GLIBCXX_REGEX_STATE_LIMIT`

#### Typedefs

- `template<typename _CharT >`  
`using std::__detail::_Matcher = std::function< bool(_CharT)>`
- `typedef long std::__detail::_StateldT`

#### Enumerations

- enum `std::__detail::_Opcode` : int {  
    `_S_opcode_unknown`, `_S_opcode_alternative`, `_S_opcode_repeat`, `_S_opcode_backref`,  
    `_S_opcode_line_begin_assertion`, `_S_opcode_line_end_assertion`, `_S_opcode_word_boundary`, `_S_↵ opcode_subexpr_lookahead`,  
    `_S_opcode_subexpr_begin`, `_S_opcode_subexpr_end`, `_S_opcode_dummy`, `_S_opcode_match`,  
    `_S_opcode_accept` }

## Variables

- static const `_StateIdT` `std::__detail::_S_invalid_state_id`

## 6.489.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

6.490 `regex_automaton.tcc` File Reference

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## 6.490.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

6.491 `regex_compiler.h` File Reference

## Classes

- struct [std::\\_\\_detail::BracketMatcher](#)`<_TraitsT, __icase, __collate >`
- struct [std::\\_\\_detail::BracketMatcher](#)`<_TraitsT, __icase, __collate >`
- class [std::\\_\\_detail::Compiler](#)`<_TraitsT >`
- class [std::regex\\_traits](#)`<_Ch_type >`

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Typedefs

- template<typename `_Iter` , typename `_TraitsT` >  
using `std::__detail::_disable_if_contiguous_normal_iter` = typename enable\_if< `!__is_contiguous_normal_iter``<_Iter >::value, std::shared_ptr``<const _NFA``<_TraitsT > > >::type`
- template<typename `_Iter` , typename `_TraitsT` >  
using `std::__detail::_enable_if_contiguous_normal_iter` = typename enable\_if< `__is_contiguous_normal_iter``<_Iter >::value, std::shared_ptr``<const _NFA``<_TraitsT > > >::type`



## Functions

- `template<typename _TraitsT, typename _Fwdlter >`  
`__enable_if_contiguous_normal_iter<_Fwdlter, _TraitsT > std::__detail::__compile_nfa (_Fwdlter __first, ↵`  
`_Fwdlter __last, const typename _TraitsT::locale_type &__loc, regex_constants::syntax_option_type __flags)`
- `template<typename _TraitsT, typename _Fwdlter >`  
`__disable_if_contiguous_normal_iter<_Fwdlter, _TraitsT > std::__detail::__compile_nfa (_Fwdlter __first, ↵`  
`_Fwdlter __last, const typename _TraitsT::locale_type &__loc, regex_constants::syntax_option_type __flags)`

### 6.491.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.492 `regex_compiler.tcc` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Macros

- `#define __INSERT_REGEX_MATCHER(__func, ...)`

### 6.492.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.493 `regex_constants.h` File Reference

### Namespaces

- [std](#)
- [std::regex\\_constants](#)

## 5.1 Regular Expression Syntax Options

- enum `std::regex_constants::__syntax_option` {  
`_S_icalse`, `_S_nosubs`, `_S_optimize`, `_S_collate`,  
`_S_ECMAScript`, `_S_basic`, `_S_extended`, `_S_awk`,  
`_S_grep`, `_S_egrep`, `_S_polynomial`, `_S_syntax_last` }
- enum `std::regex_constants::syntax_option_type` : unsigned int
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::icalse`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::nosubs`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::optimize`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::collate`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::ECMAScript`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::basic`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::extended`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::awk`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::grep`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::egrep`
- `_GLIBCXX17_INLINE` constexpr `syntax_option_type` `std::regex_constants::__polynomial`
- constexpr `syntax_option_type` `std::regex_constants::operator&` (`syntax_option_type __a`, `syntax_option_type __b`)
- constexpr `syntax_option_type` `std::regex_constants::operator|` (`syntax_option_type __a`, `syntax_option_type __b`)
- constexpr `syntax_option_type` `std::regex_constants::operator^` (`syntax_option_type __a`, `syntax_option_type __b`)
- constexpr `syntax_option_type` `std::regex_constants::operator~` (`syntax_option_type __a`)
- `syntax_option_type & std::regex_constants::operator&=` (`syntax_option_type & __a`, `syntax_option_type __b`)
- `syntax_option_type & std::regex_constants::operator|=` (`syntax_option_type & __a`, `syntax_option_type __b`)
- `syntax_option_type & std::regex_constants::operator^=` (`syntax_option_type & __a`, `syntax_option_type __b`)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `std::regex_constants::__match_flag` {  
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,  
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,  
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- enum `std::regex_constants::match_flag_type` : unsigned int
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_default`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_not_bol`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_not_eol`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_not_bow`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_not_eow`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_any`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_not_null`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_continuous`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::match_prev_avail`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::format_default`
- `_GLIBCXX17_INLINE` constexpr `match_flag_type` `std::regex_constants::format_sed`

- `_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::format_no_copy`
- `_GLIBCXX17_INLINE constexpr match_flag_type std::regex_constants::format_first_only`
- `constexpr match_flag_type std::regex_constants::operator& (match_flag_type __a, match_flag_type __b)`
- `constexpr match_flag_type std::regex_constants::operator| (match_flag_type __a, match_flag_type __b)`
- `constexpr match_flag_type std::regex_constants::operator^ (match_flag_type __a, match_flag_type __b)`
- `constexpr match_flag_type std::regex_constants::operator~ (match_flag_type __a)`
- `match_flag_type & std::regex_constants::operator&= (match_flag_type &__a, match_flag_type __b)`
- `match_flag_type & std::regex_constants::operator|= (match_flag_type &__a, match_flag_type __b)`
- `match_flag_type & std::regex_constants::operator^= (match_flag_type &__a, match_flag_type __b)`

### 6.493.1 Detailed Description

Constant definitions for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 6.494 regex\_error.h File Reference

### Classes

- class `std::regex_error`

### Namespaces

- `std`
- `std::regex_constants`

### Functions

- void `std::__throw_regex_error` (regex\_constants::error\_type \_\_ecode)
- void `std::__throw_regex_error` (regex\_constants::error\_type \_\_ecode, const char \*\_\_what)

## 5.3 Error Types

- enum `std::regex_constants::error_type` {  
`_S_error_collate, _S_error_ctype, _S_error_escape, _S_error_backref,`  
`_S_error_brack, _S_error_paren, _S_error_brace, _S_error_badbrace,`  
`_S_error_range, _S_error_space, _S_error_badrepeat, _S_error_complexity,`  
`_S_error_stack }`
- `constexpr error_type std::regex_constants::error_collate` (`_S_error_collate`)
- `constexpr error_type std::regex_constants::error_ctype` (`_S_error_ctype`)
- `constexpr error_type std::regex_constants::error_escape` (`_S_error_escape`)
- `constexpr error_type std::regex_constants::error_backref` (`_S_error_backref`)
- `constexpr error_type std::regex_constants::error_brack` (`_S_error_brack`)
- `constexpr error_type std::regex_constants::error_paren` (`_S_error_paren`)
- `constexpr error_type std::regex_constants::error_brace` (`_S_error_brace`)
- `constexpr error_type std::regex_constants::error_badbrace` (`_S_error_badbrace`)
- `constexpr error_type std::regex_constants::error_range` (`_S_error_range`)
- `constexpr error_type std::regex_constants::error_space` (`_S_error_space`)
- `constexpr error_type std::regex_constants::error_badrepeat` (`_S_error_badrepeat`)
- `constexpr error_type std::regex_constants::error_complexity` (`_S_error_complexity`)
- `constexpr error_type std::regex_constants::error_stack` (`_S_error_stack`)

## 6.494.1 Detailed Description

Error and exception objects for the `std::regex` library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

6.495 `regex_executor.h` File Reference

## Classes

- class `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode>`

## Namespaces

- `std`
- `std::__detail`

## 6.495.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

6.496 `regex_executor.tcc` File Reference

## Namespaces

- `std`
- `std::__detail`

## 6.496.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

6.497 `regex_scanner.h` File Reference

## Classes

- class `std::__detail::_Scanner<_CharT>`

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### 6.497.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 6.498 `regex_scanner.tcc` File Reference

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### 6.498.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 6.499 `resize_fn_imps.hpp` File Reference

#### 6.499.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions.

### 6.500 `resize_fn_imps.hpp` File Reference

#### 6.500.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions.

### 6.501 `resize_no_store_hash_fn_imps.hpp` File Reference

#### 6.501.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is not stored.

## 6.502 `resize_no_store_hash_fn_imps.hpp` File Reference

### 6.502.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is not stored.

## 6.503 `resize_policy.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::resize\\_policy<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.503.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 6.504 `resize_store_hash_fn_imps.hpp` File Reference

### 6.504.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is stored.

## 6.505 `resize_store_hash_fn_imps.hpp` File Reference

### 6.505.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is stored.

## 6.506 `rope` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::rope<\\_CharT, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::rope<\\_CharT, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::\\_\\_detail](#)
- [std](#)
- [std::tr1](#)

## Macros

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`
- `#define _ROPE`

## Typedefs

- `typedef rope< char > __gnu_cxx::crope`
- `typedef rope< wchar_t > __gnu_cxx::wrope`

## Enumerations

- `enum { _S_max_rope_depth }`
- `enum _Tag { _S_leaf, _S_concat, _S_substringfn, _S_function }`

## Functions

- `crope::reference __gnu_cxx::__mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void __gnu_cxx::__Destroy_const (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void __gnu_cxx::__Destroy_const (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp >)`
- `template<class _CharT >`  
`void __gnu_cxx::__S_cond_store_eos (_CharT &)`
- `void __gnu_cxx::__S_cond_store_eos (char &__c)`
- `void __gnu_cxx::__S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`  
`_CharT __gnu_cxx::__S_eos (_CharT *)`
- `template<class _CharT >`  
`bool __gnu_cxx::__S_is_basic_char_type (_CharT *)`
- `bool __gnu_cxx::__S_is_basic_char_type (char *)`
- `bool __gnu_cxx::__S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`  
`bool __gnu_cxx::__S_is_one_byte_char_type (_CharT *)`
- `bool __gnu_cxx::__S_is_one_byte_char_type (char *)`

- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`



- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

## Variables

- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

## 6.506.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.507 ropeimpl.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<class _CharT, class _Traits >`  
`void __gnu_cxx::Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`  
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`  
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

## 6.507.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

## 6.508 rotate\_fn\_imps.hpp File Reference

## 6.508.1 Detailed Description

Contains imps for rotating nodes.

## 6.509 rotate\_fn\_imps.hpp File Reference

## 6.509.1 Detailed Description

Contains imps for rotating nodes.

## 6.510 `safe_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence\\_base](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### Functions

- bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const [\\_\\_Safe\\_iterator\\_base](#) \* \_\_x)

#### 6.510.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.511 `safe_container.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_container<\\_SafeContainer, \\_Alloc, \\_SafeBase, \\_IsCxx11AllocatorAware >](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

#### 6.511.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.512 `safe_iterator.h` File Reference

### Classes

- struct [\\_\\_gnu\\_debug::\\_\\_BeforeBeginHelper<\\_Sequence >](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Iterator, \\_Sequence >](#)
- struct [\\_\\_gnu\\_debug::\\_\\_Sequence\\_traits<\\_Sequence >](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Functions

- `template<typename _Iterator, typename _Sequence >  
_Iterator __gnu_debug::__base (const _Safe_iterator< _Iterator, _Sequence > &__it, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence >  
const _Safe_iterator< _Iterator, _Sequence > & __gnu_debug::__base (const _Safe_iterator< _Iterator, \_↵  
_Sequence > &__it, std::input\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence >  
auto __gnu_debug::__base (const _Safe_iterator< _Iterator, _Sequence > &__it) -> decltype(__base(\_↵  
__it, std::iterator\_category(__it)))`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _Iterator, typename _Sequence >  
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const _Safe_iterator< _Iterator, \_↵  
_Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence >  
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const _Safe_iterator< _Iterator, \_↵  
_Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, std::input\_iterator\_tag)`
- `template<typename _Iterator, typename _Sequence >  
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance_from_begin (const _Safe_iterator< \_↵  
_Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence >  
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance_to_end (const _Safe_iterator< _Iterator,  
_Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence >  
_Iterator __gnu_debug::__unsafe (const _Safe_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator<  
_Iterator, _Sequence > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::operator!= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<  
_IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::operator!= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator<  
_Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >  
_Safe_iterator< _Iterator, _Sequence > __gnu_debug::operator+ (typename _Safe_iterator< _Iterator, \_↵  
_Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
_Safe_iterator< _IteratorL, _Sequence >::difference_type __gnu_debug::operator- (const _Safe_iterator< \_↵  
_IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >  
_Safe_iterator< _Iterator, _Sequence >::difference_type __gnu_debug::operator- (const _Safe_iterator< \_↵  
_Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool __gnu_debug::operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator<  
_IteratorR, _Sequence > &__rhs) noexcept`

- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`

#### 6.512.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 6.513 safe\_iterator.tcc File Reference

#### Namespaces

- [\\_\\_gnu\\_debug](#)

#### Macros

- `#define _GLIBCXX_DEBUG_SAFE_ITERATOR_TCC`

#### 6.513.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.514 `safe_local_iterator.h` File Reference

## Classes

- class [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_Iterator, \\_Sequence>](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Functions

- template<typename \_Iterator, typename \_Sequence>  
bool [\\_\\_gnu\\_debug::check\\_dereferenceable](#) (const \_Safe\_local\_iterator<\_Iterator, \_Sequence> &\_\_x)
- template<typename \_Iterator, typename \_Sequence>  
[std::pair](#)<typename std::iterator\_traits<\_Iterator>::difference\_type, \_Distance\_precision> [\\_\\_gnu\\_debug::get\\_distance](#)  
(const \_Safe\_local\_iterator<\_Iterator, \_Sequence> &\_\_first, const \_Safe\_local\_iterator<\_Iterator, \_Sequence>  
> &\_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_Iterator, typename \_Sequence>  
\_Iterator [\\_\\_gnu\\_debug::unsafe](#) (const \_Safe\_local\_iterator<\_Iterator, \_Sequence> &\_\_it)
- template<typename \_Iterator, typename \_Sequence>  
bool [\\_\\_gnu\\_debug::valid\\_range](#) (const \_Safe\_local\_iterator<\_Iterator, \_Sequence> &\_\_first, const \_Safe\_↵  
local\_iterator<\_Iterator, \_Sequence> &\_\_last, typename \_Distance\_traits<\_Iterator>::\_\_type &\_\_dist\_info)
- template<typename \_IteratorL, typename \_IteratorR, typename \_Sequence>  
bool [\\_\\_gnu\\_debug::operator!=](#) (const \_Safe\_local\_iterator<\_IteratorL, \_Sequence> &\_\_lhs, const \_Safe\_↵  
local\_iterator<\_IteratorR, \_Sequence> &\_\_rhs)
- template<typename \_Iterator, typename \_Sequence>  
bool [\\_\\_gnu\\_debug::operator!=](#) (const \_Safe\_local\_iterator<\_Iterator, \_Sequence> &\_\_lhs, const \_Safe\_↵  
local\_iterator<\_Iterator, \_Sequence> &\_\_rhs)
- template<typename \_IteratorL, typename \_IteratorR, typename \_Sequence>  
bool [\\_\\_gnu\\_debug::operator==](#) (const \_Safe\_local\_iterator<\_IteratorL, \_Sequence> &\_\_lhs, const \_Safe\_↵  
local\_iterator<\_IteratorR, \_Sequence> &\_\_rhs)
- template<typename \_Iterator, typename \_Sequence>  
bool [\\_\\_gnu\\_debug::operator==](#) (const \_Safe\_local\_iterator<\_Iterator, \_Sequence> &\_\_lhs, const \_Safe\_↵  
local\_iterator<\_Iterator, \_Sequence> &\_\_rhs)

## 6.514.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.515 `safe_local_iterator.tcc` File Reference

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_SAFE_LOCAL_ITERATOR_TCC`

### 6.515.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.516 `safe_sequence.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::After\\_nth\\_from<\\_Iterator>](#)
- class [\\_\\_gnu\\_debug::Equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::Not\\_equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::Safe\\_node\\_sequence<\\_Sequence>](#)
- class [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### 6.516.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.517 `safe_sequence.tcc` File Reference

### Namespaces

- [\\_\\_gnu\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_SAFE_SEQUENCE_TCC`

### 6.517.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.518 `safe_unordered_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container\\_base](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

#### 6.518.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.519 `safe_unordered_container.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_unordered\\_container<\\_Container>](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

#### 6.519.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.520 `safe_unordered_container.tcc` File Reference

### Namespaces

- [\\_\\_gnu\\_debug](#)

### Macros

- `#define \_GLIBCXX\_DEBUG\_SAFE\_UNORDERED\_CONTAINER\_TCC`

#### 6.520.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.



## 6.521 `sample_probe_fn.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_probe\\_fn](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.521.1 Detailed Description

Contains a sample probe policy.

## 6.522 `sample_range_hashing.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_range\\_hashing](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.522.1 Detailed Description

Contains a range hashing policy.

## 6.523 `sample_ranged_hash_fn.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_hash\\_fn](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.523.1 Detailed Description

Contains a ranged hash policy.

## 6.524 sample\_ranged\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_probe\\_fn](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.524.1 Detailed Description

Contains a ranged probe policy.

## 6.525 sample\_resize\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_policy](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.525.1 Detailed Description

Contains a sample resize policy for hash tables.

## 6.526 sample\_resize\_trigger.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_trigger](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.526.1 Detailed Description

Contains a sample resize trigger policy class.

## 6.527 `sample_size_policy.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_size\\_policy](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.527.1 Detailed Description

Contains a sample size resize-policy.

## 6.528 `sample_tree_node_update.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_tree\\_node\\_update< Const\\_Node\\_Iter, Node\\_Iter, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.528.1 Detailed Description

Contains a samle node update functor.

## 6.529 `sample_trie_access_traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.529.1 Detailed Description

Contains a sample probe policy.

## 6.530 sample\_trie\_node\_update.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_trie\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.530.1 Detailed Description

Contains a samle node update functor.

## 6.531 sample\_update\_policy.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_update\\_policy](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.531.1 Detailed Description

Contains a sample policy for list update containers.

## 6.532 scoped\_allocator File Reference

### Classes

- class [std::scoped\\_allocator\\_adaptor](#)< \_OuterAlloc, \_InnerAllocs >
- class [std::scoped\\_allocator\\_adaptor](#)< \_OuterAlloc, \_InnerAllocs >

### Namespaces

- [std](#)

### Macros

- `#define \_SCOPED\_ALLOCATOR`

## Typedefs

- `template<typename _Alloc >`  
`using std::__outer_allocator_t = decltype(std::declval<_Alloc >().outer_allocator())`

## Functions

- `template<typename _Alloc >`  
`__outermost_type<_Alloc >::type & std::__outermost (_Alloc &__a)`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool std::operator!= (const scoped_allocator_adaptor<_OutA1, _InA... > &__a, const scoped_allocator_↵  
adaptor<_OutA2, _InA... > &__b) noexcept`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool std::operator== (const scoped_allocator_adaptor<_OutA1, _InA... > &__a, const scoped_allocator_↵  
adaptor<_OutA2, _InA... > &__b) noexcept`

### 6.532.1 Detailed Description

This is a Standard C++ Library header.

## 6.533 `search.h` File Reference

### Namespaces

- [`\_\_gnu\_parallel`](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp >`  
`void \_\_gnu\_parallel::\_\_calc\_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp * __off)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`\_\_RAIter1 \_\_gnu\_parallel::\_\_search\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, ↵  
_RAIter2 __end2, _Pred __pred)`

### 6.533.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.534 `set` File Reference

### Macros

- `#define _GLIBCXX_SET`

## 6.534.1 Detailed Description

This is a Standard C++ Library header.

## 6.535 set File Reference

## Macros

- `#define _GLIBCXX_DEBUG_SET`

## 6.535.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.536 set File Reference

## Macros

- `#define _GLIBCXX_PROFILE_SET`

## 6.536.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.537 set File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_SET`

## Typedefs

- `template<typename _Key , typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::multiset = std::multiset< _Key, _Compare, polymorphic_allocator< _Key > >`
- `template<typename _Key , typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::set = std::set< _Key, _Compare, polymorphic_allocator< _Key > >`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`

### 6.537.1 Detailed Description

This is a TS C++ Library header.

## 6.538 set.h File Reference

### Classes

- class `std::\_\_debug::set< \_Key, \_Compare, \_Allocator >`

### Namespaces

- `std`
- `std::\_\_debug`

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`  
`noexcept(/*conditional */)`

## 6.538.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.539 set.h File Reference

## Classes

- class [std::\\_\\_profile::set<\\_Key, \\_Compare, \\_Allocator>](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)  
noexcept(/*conditional */)`

## 6.539.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.540 set\_operations.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)



## Functions

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b, std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`

### 6.540.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 6.541 settings.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Settings](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_PARALLEL\_CONDITION(__c)`

### 6.541.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

## 6.541.2 parallelization\_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and \_\_off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel↵::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

## 6.541.3 Macro Definition Documentation

## 6.541.3.1 \_GLIBCXX\_PARALLEL\_CONDITION

```
#define _GLIBCXX_PARALLEL_CONDITION(
    __c )
```

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

## Parameters

<code>↵</code>	A condition that is convertible to bool that is overruled by <code>__gnu_parallel::_Settings::algorithm_strategy</code> .
<code>__c</code>	Usually a decision based on the input size.

Definition at line 95 of file settings.h.

## 6.542 `shared_mutex` File Reference

### Classes

- class [std::\\_\\_shared\\_mutex\\_cv](#)
- class [std::shared\\_lock< \\_Mutex >](#)
- class [std::shared\\_timed\\_mutex](#)

### Namespaces

- [std](#)

### Macros

- `#define \_GLIBCXX\_SHARED\_MUTEX`
- `#define \_\_cpp\_lib\_shared\_timed\_mutex`
- `using std::\_\_shared\_timed\_mutex\_base = \_\_shared\_mutex\_cv`
- `template<typename \_Mutex >`  
`void std::swap (shared\_lock< \_Mutex > &__x, shared\_lock< \_Mutex > &__y) noexcept`

### 6.542.1 Detailed Description

This is a Standard C++ Library header.

## 6.543 `shared_ptr.h` File Reference

### Classes

- class [std::enable\\_shared\\_from\\_this< \\_Tp >](#)
- struct [std::hash< \[shared\\\_ptr< \\\_Tp >\]\(#\) >](#)
- struct [std::owner\\_less< \\_Tp >](#)
- struct [std::owner\\_less< \[shared\\\_ptr< \\\_Tp >\]\(#\) >](#)
- struct [std::owner\\_less< void >](#)
- struct [std::owner\\_less< \[weak\\\_ptr< \\\_Tp >\]\(#\) >](#)
- class [std::shared\\_ptr< \\_Tp >](#)
- class [std::weak\\_ptr< \\_Tp >](#)

### Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_enable_shared_from_this`

## Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr<_Tp> std::allocate\_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename _Up >`  
`shared_ptr<_Tp> std::const\_pointer\_cast (const shared_ptr<_Up> &__r) noexcept`
- `template<typename _Tp, typename _Up >`  
`shared_ptr<_Tp> std::dynamic\_pointer\_cast (const shared_ptr<_Up> &__r) noexcept`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * std::get\_deleter (const __shared_ptr<_Tp, _Lp> &__p) noexcept`
- `template<typename _Del, typename _Tp >`  
`_Del * std::get\_deleter (const shared_ptr<_Tp> &__p) noexcept`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr<_Tp> std::make\_shared (_Args &&... __args)`
- `template<typename _Tp, typename _Up >`  
`bool std::operator!= (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator!= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator!= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::operator< (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator< (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator< (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic\_ostream<_Ch, _Tr> & std::operator<< (std::basic\_ostream<_Ch, _Tr> &__os, const __shared_ptr<_Tp, _Lp> &__p)`
- `template<typename _Tp, typename _Up >`  
`bool std::operator<= (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator<= (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator<= (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::operator== (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator== (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator== (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::operator> (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator> (const shared_ptr<_Tp> &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator> (nullptr_t, const shared_ptr<_Tp> &__a) noexcept`

- `template<typename _Tp, typename _Up >`  
`bool std::operator>= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp >`  
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`  
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

#### 6.543.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

#### 6.544 `shared_ptr.h` File Reference

##### Classes

- `struct std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >`
- `struct std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >`
- `struct std::hash< experimental::shared\_ptr< \_Tp > >`

##### Namespaces

- `std`

##### Functions

- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __↵ failure)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::atomic_compare_exchange_weak (shared_ptr< _Tp > *__↵ p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__↵ p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`

- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_exchange_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order __mo)`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_store_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Del, typename _Tp >`  
`_Del * std::experimental::fundamentals_v2::get_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp >`  
`std::basic\_ostream< _Ch, _Tr > & std::experimental::fundamentals_v2::operator<< (std::basic\_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`

- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp , typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::reinterpret_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp , typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

## Variables

- `template<typename _Yp , typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::__sp_compatible_v`
- `template<typename _Tp , typename _Yp >`  
`constexpr bool std::experimental::fundamentals_v2::__sp_is_constructible_v`

### 6.544.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/memory>`.

### 6.544.2 Function Documentation

## 6.544.2.1 get\_deleter()

```
template<typename _Del , typename _Tp >
_Del* std::experimental::fundamentals_v2::get_deleter (
    const shared_ptr< _Tp > & __p ) [inline], [noexcept]
```

C++14 §20.8.2.2.10.

Definition at line 486 of file experimental/bits/shared\_ptr.h.

## 6.545 shared\_ptr\_atomic.h File Reference

## Namespaces

- [std](#)

## Functions

- template<typename \_Tp , \_Lock\_policy \_Lp>  
bool [std::atomic\\_is\\_lock\\_free](#) (const \_\_shared\_ptr< \_Tp, \_Lp > \*\_\_p)
- template<typename \_Tp >  
bool [std::atomic\\_is\\_lock\\_free](#) (const shared\_ptr< \_Tp > \*\_\_p)
- template<typename \_Tp >  
shared\_ptr< \_Tp > [std::atomic\\_load\\_explicit](#) (const shared\_ptr< \_Tp > \*\_\_p, memory\_order)
- template<typename \_Tp >  
shared\_ptr< \_Tp > [std::atomic\\_load](#) (const shared\_ptr< \_Tp > \*\_\_p)
- template<typename \_Tp , \_Lock\_policy \_Lp>  
\_\_shared\_ptr< \_Tp, \_Lp > [std::atomic\\_load\\_explicit](#) (const \_\_shared\_ptr< \_Tp, \_Lp > \*\_\_p, memory\_order)
- template<typename \_Tp , \_Lock\_policy \_Lp>  
\_\_shared\_ptr< \_Tp, \_Lp > [std::atomic\\_load](#) (const \_\_shared\_ptr< \_Tp, \_Lp > \*\_\_p)
- template<typename \_Tp >  
void [std::atomic\\_store\\_explicit](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r, memory\_order)
- template<typename \_Tp >  
void [std::atomic\\_store](#) (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r)
- template<typename \_Tp , \_Lock\_policy \_Lp>  
void [std::atomic\\_store\\_explicit](#) (\_\_shared\_ptr< \_Tp, \_Lp > \*\_\_p, \_\_shared\_ptr< \_Tp, \_Lp > \_\_r, memory\_order)
- template<typename \_Tp , \_Lock\_policy \_Lp>  
void [std::atomic\\_store](#) (\_\_shared\_ptr< \_Tp, \_Lp > \*\_\_p, \_\_shared\_ptr< \_Tp, \_Lp > \_\_r)



- `template<typename _Tp >`  
`shared_ptr<_Tp> std::atomic_exchange_explicit (shared_ptr<_Tp> *__p, shared_ptr<_Tp> __r, memory_order)`
- `template<typename _Tp >`  
`shared_ptr<_Tp> std::atomic_exchange (shared_ptr<_Tp> *__p, shared_ptr<_Tp> __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr<_Tp, _Lp> std::atomic_exchange_explicit (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr<_Tp, _Lp> std::atomic_exchange (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> __r)`
  
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_strong_explicit (shared_ptr<_Tp> *__p, shared_ptr<_Tp> *__v, shared_ptr<_Tp> __w, memory_order, memory_order)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_strong (shared_ptr<_Tp> *__p, shared_ptr<_Tp> *__v, shared_ptr<_Tp> __w)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_weak_explicit (shared_ptr<_Tp> *__p, shared_ptr<_Tp> *__v, shared_ptr<_Tp> __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp >`  
`bool std::atomic_compare_exchange_weak (shared_ptr<_Tp> *__p, shared_ptr<_Tp> *__v, shared_ptr<_Tp> __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong_explicit (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> *__v, __shared_ptr<_Tp, _Lp> __w, memory_order, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> *__v, __shared_ptr<_Tp, _Lp> __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak_explicit (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> *__v, __shared_ptr<_Tp, _Lp> __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak (__shared_ptr<_Tp, _Lp> *__p, __shared_ptr<_Tp, _Lp> *__v, __shared_ptr<_Tp, _Lp> __w)`

### 6.545.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.546 shared\_ptr\_base.h File Reference

## Classes

- struct [std::\\_Sp\\_ebo\\_helper<\\_Nm, \\_Tp, false>](#)
- struct [std::\\_Sp\\_ebo\\_helper<\\_Nm, \\_Tp, true>](#)
- class [std::bad\\_weak\\_ptr](#)
- class [std::enable\\_shared\\_from\\_this<\\_Tp>](#)
- struct [std::hash<\\_\\_shared\\_ptr<\\_Tp, \\_Lp>>](#)
- struct [std::owner\\_less<\\_Tp>](#)
- class [std::shared\\_ptr<\\_Tp>](#)
- class [std::weak\\_ptr<\\_Tp>](#)

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_shared_ptr_arrays`

## Functions

- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>  
__shared_ptr<_Tp, _Lp> std::__allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>  
__shared_ptr<_Tp, _Lp> std::__make_shared (_Args &&... __args)`
- `void std::__throw_bad_weak_ptr ()`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr<_Tp, _Lp> std::const_pointer_cast (const __shared_ptr<_Tp1, _Lp> &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (const __shared_ptr<_Tp1, _Lp> &__r) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator!= (const __shared_ptr<_Tp1, _Lp> &__a, const __shared_ptr<_Tp2, _Lp> &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator!= (const __shared_ptr<_Tp, _Lp> &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator!= (nullptr_t, const __shared_ptr<_Tp, _Lp> &__a) noexcept`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>  
bool std::operator< (const __shared_ptr<_Tp, _Lp> &__a, const __shared_ptr<_Up, _Lp> &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator< (const __shared_ptr<_Tp, _Lp> &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator< (nullptr_t, const __shared_ptr<_Tp, _Lp> &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator<= (const __shared_ptr<_Tp1, _Lp> &__a, const __shared_ptr<_Tp2, _Lp> &__b) noexcept`

- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool std::operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool std::operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`

#### 6.546.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

### 6.547 `size_fn_imps.hpp` File Reference

#### 6.547.1 Detailed Description

Contains implementations of `cc_ht_map_'s` entire container size related functions.

### 6.548 `slice_array.h` File Reference

#### Classes

- class `std::slice`
- class `std::slice_array< _Tp >`

## Namespaces

- [std](#)

## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 6.548.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.549 slist File Reference

## Classes

- class [\\_\\_gnu\\_cxx::slist<\\_Tp, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define _SLIST`

## Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __gnu_cxx::__slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __gnu_cxx::__slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __node)`
- `_Slist_node_base * __gnu_cxx::__slist_reverse (_Slist_node_base * __node)`
- `size_t __gnu_cxx::__slist_size (_Slist_node_base * __node)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __before_last)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<class _Tp, class _Alloc>  
bool __gnu_cxx::operator!= (const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<class _Tp, class _Alloc>  
bool __gnu_cxx::operator< (const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<class _Tp, class _Alloc>  
bool __gnu_cxx::operator<= (const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<class _Tp, class _Alloc>  
bool __gnu_cxx::operator== (const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<class _Tp, class _Alloc>  
bool __gnu_cxx::operator> (const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<class _Tp, class _Alloc>  
bool __gnu_cxx::operator>= (const slist<_Tp, _Alloc> &_SL1, const slist<_Tp, _Alloc> &_SL2)`
- `template<class _Tp, class _Alloc>  
void __gnu_cxx::swap (slist<_Tp, _Alloc> &_x, slist<_Tp, _Alloc> &_y)`

### 6.549.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 6.550 sort.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↵  
parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵  
_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵  
_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵  
_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __↵  
parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort↵  
_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag  
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __↵  
parallelism)`

### 6.550.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

## 6.551 specfun.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define __cpp_lib_math_special_functions`
- `#define` `__STDCPP_MATH_SPEC_FUNCS__`

## Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)`
- `float __gnu_cxx::airy_aif (float __x)`
- `long double __gnu_cxx::airy_ail (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)`
- `float __gnu_cxx::airy_bif (float __x)`
- `long double __gnu_cxx::airy_bil (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpa, typename _Tpb >`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)`
- `float std::betaf (float __a, float __b)`
- `long double std::betal (long double __a, long double __b)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)`
- `float std::comp_ellint_1f (float __k)`
- `long double std::comp_ellint_1l (long double __k)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)`
- `long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_if (float __nu, float __x)`
- `long double std::cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_jf (float __nu, float __x)`
- `long double std::cyl_bessel_jl (long double __nu, long double __x)`

- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_kf (float __nu, float __x)`
- `long double std::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_neumannf (float __nu, float __x)`
- `long double std::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_1f (float __k, float __phi)`
- `long double std::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_2f (float __k, float __phi)`
- `long double std::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float std::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::expint ( _Tp __x)`
- `float std::expintf (float __x)`
- `long double std::expintl (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)`
- `float std::hermitef (unsigned int __n, float __x)`
- `long double std::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)`
- `long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)`
- `float std::laguerref (unsigned int __n, float __x)`
- `long double std::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)`
- `float std::legendref (unsigned int __l, float __x)`
- `long double std::legendrel (unsigned int __l, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta ( _Tp __s)`
- `float std::riemann_zetaf (float __s)`
- `long double std::riemann_zetal (long double __s)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)`
- `float std::sph_besself (unsigned int __n, float __x)`
- `long double std::sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)`

- long double [std::sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)

#### 6.551.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>`.

## 6.552 splay\_fn\_imps.hpp File Reference

#### 6.552.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 6.553 splay\_tree\_.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_S_TREE_BASE`
- `#define PB_DS_S_TREE_BASE_NAME`
- `#define PB_DS_S_TREE_NAME`

#### 6.553.1 Detailed Description

Contains an implementation class for splay trees.



## 6.554 `split_fn_imps.hpp` File Reference

### 6.554.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 6.555 `split_join_fn_imps.hpp` File Reference

### 6.555.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 6.556 `split_join_fn_imps.hpp` File Reference

### 6.556.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 6.557 `split_join_fn_imps.hpp` File Reference

### 6.557.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 6.558 `split_join_fn_imps.hpp` File Reference

### 6.558.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 6.559 `split_join_fn_imps.hpp` File Reference

### 6.559.1 Detailed Description

Contains an implementation class for a pairing heap.

## 6.560 split\_join\_fn\_imps.hpp File Reference

### 6.560.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 6.561 split\_join\_fn\_imps.hpp File Reference

### 6.561.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 6.562 split\_join\_fn\_imps.hpp File Reference

### 6.562.1 Detailed Description

Contains an implementation class for splay\_tree\_.

## 6.563 split\_join\_fn\_imps.hpp File Reference

### 6.563.1 Detailed Description

Contains an implementation for thin\_heap\_.

## 6.564 sso\_string\_base.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 6.564.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.565 sstream File Reference

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_SSTREAM`

## Functions

- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_istream< _CharT, _Traits, _Allocator > &__x, basic_istream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)`

### 6.565.1 Detailed Description

This is a Standard C++ Library header.

## 6.566 sstream.tcc File Reference

## Namespaces

- [std](#)

## Macros

- `#define _SSTREAM_TCC`

### 6.566.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

## 6.567 stack File Reference

### Macros

- `#define _GLIBCXX_STACK`

#### 6.567.1 Detailed Description

This is a Standard C++ Library header.

## 6.568 standard\_policies.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::default\\_comb\\_hash\\_fn](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_eq\\_fn< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_hash\\_fn< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_probe\\_fn< Comb\\_Probe\\_Fn >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_resize\\_policy< Comb\\_Hash\\_Fn >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< std::basic\\_string< Char, Char\\_Traits, std::allocator< char > > >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_update\\_policy](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

### Enumerations

- enum { **default\_store\_hash** }

#### 6.568.1 Detailed Description

Contains standard policies for containers.

#### 6.568.2 Enumeration Type Documentation

### 6.568.2.1 anonymous enum

`anonymous enum`

Enumeration for default behavior of stored hash data.

Definition at line 74 of file `standard_policies.hpp`.

## 6.569 std\_abs.h File Reference

### Namespaces

- [std](#)

### Functions

- long **std::abs** (long \_\_i)
- long long **std::abs** (long long \_\_x)
- constexpr double **std::abs** (double \_\_x)
- constexpr float **std::abs** (float \_\_x)
- constexpr long double **std::abs** (long double \_\_x)

### 6.569.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>` or `<cstdlib>`.

## 6.570 std\_function.h File Reference

### Classes

- struct [std::\\_\\_is\\_location\\_invariant<\\_Tp>](#)
- class [std::\\_Function\\_base](#)
- class [std::bad\\_function\\_call](#)
- class [std::function<\\_Res\(\\_ArgTypes...\)>](#)

### Namespaces

- [std](#)

### Typedefs

- `template<typename _From, typename _To>  
using std::__check_func_return_type = __or_< is_void<_To>, is_same<_From, _To>, is_convertible<_From, _To>>>`

## Enumerations

- enum `_Manager_operation` { `__get_type_info`, `__get_functor_ptr`, `__clone_functor`, `__destroy_functor` }

## Functions

- template<typename `_Res` , typename... `_Args`>  
bool `std::operator!=` (const function< `_Res(_Args...)`> &`__f`, nullptr\_t) noexcept
- template<typename `_Res` , typename... `_Args`>  
bool `std::operator!=` (nullptr\_t, const function< `_Res(_Args...)`> &`__f`) noexcept
- template<typename `_Res` , typename... `_Args`>  
bool `std::operator==` (const function< `_Res(_Args...)`> &`__f`, nullptr\_t) noexcept
- template<typename `_Res` , typename... `_Args`>  
bool `std::operator==` (nullptr\_t, const function< `_Res(_Args...)`> &`__f`) noexcept
- template<typename `_Res` , typename... `_Args`>  
void `std::swap` (function< `_Res(_Args...)`> &`__x`, function< `_Res(_Args...)`> &`__y`) noexcept

## 6.570.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.571 std\_mutex.h File Reference

## Classes

- struct `std::adopt_lock_t`
- struct `std::defer_lock_t`
- class `std::lock_guard<_Mutex >`
- class `std::mutex`
- struct `std::try_to_lock_t`
- class `std::unique_lock<_Mutex >`

## Namespaces

- `std`

## Functions

- template<typename `_Mutex` >  
void `std::swap` (unique\_lock< `_Mutex` > &`__x`, unique\_lock< `_Mutex` > &`__y`) noexcept

## Variables

- `_GLIBCXX17_INLINE` constexpr adopt\_lock\_t `std::adopt_lock`
- `_GLIBCXX17_INLINE` constexpr defer\_lock\_t `std::defer_lock`
- `_GLIBCXX17_INLINE` constexpr try\_to\_lock\_t `std::try_to_lock`

#### 6.571.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

### 6.572 `stdc++.h` File Reference

#### 6.572.1 Detailed Description

This is an implementation file for a precompiled header.

### 6.573 `stdexcept` File Reference

#### Classes

- class [std::domain\\_error](#)
- class [std::invalid\\_argument](#)
- class [std::length\\_error](#)
- class [std::logic\\_error](#)
- class [std::out\\_of\\_range](#)
- class [std::overflow\\_error](#)
- class [std::range\\_error](#)
- class [std::runtime\\_error](#)
- class [std::underflow\\_error](#)

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_STDEXCEPT`

#### Typedefs

- `typedef basic_string< char > std::__cow_string`
- `typedef basic_string< char > std::__sso_string`

#### 6.573.1 Detailed Description

This is a Standard C++ Library header.

## 6.574 `stdio_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

#### 6.574.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.575 `stdio_sync_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

#### 6.575.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.576 `stdlib.h` File Reference

#### 6.576.1 Detailed Description

This is a Standard C++ Library header.

## 6.577 `stdtr1c++.h` File Reference

#### 6.577.1 Detailed Description

This is an implementation file for a precompiled header.



## 6.578 std\_algo.h File Reference

### Namespaces

- [std](#)

### Enumerations

- `enum { _S_threshold }`
- `enum { _S_chunk_size }`

### Functions

- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void std::chunk_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance`  
`__chunk_size, _Compare __comp)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_↵`  
`access_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, ↵`  
`_Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp, typename _CompareItTp, typename _CompareTpIt >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __↵`  
`last, const _Tp &__val, _CompareItTp __comp_it_val, _CompareTpIt __comp_val_it)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵`  
`comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↵`  
`__first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >`  
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, ↵`  
`_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag,`  
`_BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator std::find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, ↵`  
`Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iterator, typename _Predicate >`  
`_Iterator std::find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate, typename _Distance >`  
`_InputIterator std::__find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)`
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement std::__gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _IntType, typename _UniformRandomBitGenerator >`  
`pair< _IntType, _IntType > std::__gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBitGenerator &&__g)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::__includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::__inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void std::__introsort (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::__is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::__is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::__is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::__max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`  
`void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`  
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`  
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`  
`void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`

- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::__min_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > std::__minmax_element (_ForwardIterator`  
`__first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iterator, typename _Compare >`  
`void std::__move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator`  
`__last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2`  
`__last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`  
`void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2`  
`__first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::__next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::__partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator`  
`__result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`_BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::__prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::__remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator std::__replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::V2::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`_BidirectionalIterator std::V2::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`

- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::V2::rotate ( _RandomAccessIterator __first, _RandomAccessIterator __middle,`  
`_RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`  
`_BidirectionalIterator1 std::rotate_adaptive ( _BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, ↵`  
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance`  
`__buffer_size)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator >`  
`_RandomAccessIterator std::sample ( _InputIterator __first, _InputIterator __last, input_iterator_tag, ↵`  
`_RandomAccessIterator __out, random_access_iterator_tag, _Size __n, _UniformRandomBitGenerator &&g)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↵`  
`Generator >`  
`_OutputIterator std::sample ( _ForwardIterator __first, _ForwardIterator __last, forward_iterator_tag, _Output↵`  
`Iterator __out, _Cat, _Size __n, _UniformRandomBitGenerator &&g)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::search ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵`  
`first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`  
`_ForwardIterator std::search_n ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, _Unary↵`  
`Predicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`  
`_ForwardIterator std::search_n_aux ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, ↵`  
`_UnaryPredicate __unary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate >`  
`_RandomAccessIter std::search_n_aux ( _RandomAccessIter __first, _RandomAccessIter __last, _Integer ↵`  
`__count, _UnaryPredicate __unary_pred, std::random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set_intersection ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable_partition ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`  
`_ForwardIterator std::stable_partition_adaptive ( _ForwardIterator __first, _ForwardIterator __last, _Predicate`  
`__pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`  
`void std::stable_sort_adaptive ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer ↵`  
`__buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::unguarded_insertion_sort ( _RandomAccessIterator __first, _RandomAccessIterator __last, ↵`  
`_Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵  
first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵  
first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`_Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate\_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __↵  
last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __↵  
last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵  
last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵  
last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _↵  
BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _↵  
ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _↵  
ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _↵  
_Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __↵  
__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __↵  
__last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵  
Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵  
Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __↵  
__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __↵  
__last, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b,  
_Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator  
__first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator  
__first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`



- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`_OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`



- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new, ↵`  
`value)`
- `template<typename _BidirectionalIterator >`  
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator ↵`  
`__result)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::V2::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, ↵`  
`_OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp`  
`&__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp`  
`&__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 ↵`  
`__first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 ↵`  
`__first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, Input↵`  
`Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, Input↵`  
`Iterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber↵`  
`Generator &&__g)`

- `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

### 6.578.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

### 6.578.2 Function Documentation

#### 6.578.2.1 \_\_rotate() [1/3]

```
template<typename _ForwardIterator >
_FowardIterator std::_V2::__rotate (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last,
    forward_iterator_tag )
```

This is a helper function for the rotate algorithm.

Definition at line 1249 of file `stl_algo.h`.

References `std::_V2::__rotate()`, and `std::iter_swap()`.

Referenced by `std::_V2::__rotate()`.

#### 6.578.2.2 `__rotate()` [2/3]

```
template<typename _BidirectionalIterator >
_BidirectionalIterator std::_V2::__rotate (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    bidirectional_iterator_tag )
```

This is a helper function for the rotate algorithm.

Definition at line 1290 of file `stl_algo.h`.

References `std::_V2::__rotate()`.

#### 6.578.2.3 `__rotate()` [3/3]

```
template<typename _RandomAccessIterator >
_RandomAccessIterator std::_V2::__rotate (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last,
    random_access_iterator_tag )
```

This is a helper function for the rotate algorithm.

Definition at line 1328 of file `stl_algo.h`.

References `std::_V2::__rotate()`.

### 6.579 `stl_algobase.h` File Reference

#### Classes

- struct `std::char_traits<_CharT>`
- class `std::istreambuf_iterator<_CharT, _Traits>`
- class `std::ostreambuf_iterator<_CharT, _Traits>`

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_robust_nonmodifying_seq_ops`
- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`

## Functions

- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI std::__copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT >`  
`> >::__type std::__copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> >::__type std::__copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_`  
`traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`  
`(istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT`  
`> >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI std::__copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::__copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2 >`  
`bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _BinaryPredicate >`  
`bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, void >::__type std::__fill_a (_ForwardIterator __first,`  
`_ForwardIterator __last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, void >::__type std::__fill_a (_ForwardIterator __first,`  
`_ForwardIterator __last, const _Tp & __value)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::__fill_a (_Tp * __first, _Tp * __last,`  
`const _Tp & __c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a (_Output`  
`Iterator __first, _Size __n, const _Tp & __value)`

- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a ( _OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type std::__fill_n_a ( _Tp * __first, _Size __n, const _Tp &__c)`
- `template<typename _II1, typename _II2 >`  
`bool std::__lexicographical_compare_aux ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::__lexicographical_compare_impl ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `constexpr int std::__lg (int __n)`
- `constexpr unsigned std::__lg (unsigned __n)`
- `constexpr long std::__lg (long __n)`
- `constexpr unsigned long std::__lg (unsigned long __n)`
- `constexpr long long std::__lg (long long __n)`
- `constexpr unsigned long long std::__lg (unsigned long long __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::__lower_bound ( _ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::__mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::__mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator >`  
`_Iterator std::__niter_base ( _Iterator __it)`
- `template<typename _II, typename _OI >`  
`_OI std::__copy ( _II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::__copy_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`  
`bool std::__equal ( _Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`bool std::__equal ( _II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2 >`  
`bool std::__equal ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`  
`bool std::__equal ( _Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _Iiter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::__fill ( _ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::__fill_n ( _OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::__iter_swap ( _ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2 >`  
`bool std::__lexicographical_compare ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::__lexicographical_compare ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::__lower_bound ( _ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`

- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`_GLIBCXX14_CONSTEXPR const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 6.579.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 6.580 `std_bvector.h` File Reference

### Classes

- struct [std::hash<::vector< bool, \\_Alloc > >](#)
- class [std::vector< bool, \\_Alloc >](#)

### Namespaces

- [std](#)

### Typedefs

- typedef unsigned long [std::\\_Bit\\_type](#)

## Enumerations

- enum { **\_S\_word\_bit** }

## Functions

- void **std::fill\_bvector** (\_Bit\_type \* \_\_v, unsigned int \_\_first, unsigned int \_\_last, bool \_\_x)
- void **std::fill** (\_Bit\_iterator \_\_first, \_Bit\_iterator \_\_last, const bool &\_\_x)
- \_Bit\_iterator **std::operator+** (ptrdiff\_t \_\_n, const \_Bit\_iterator &\_\_x)
- \_Bit\_const\_iterator **std::operator+** (ptrdiff\_t \_\_n, const \_Bit\_const\_iterator &\_\_x)
- ptrdiff\_t **std::operator-** (const \_Bit\_iterator\_base &\_\_x, const \_Bit\_iterator\_base &\_\_y)
- void **std::swap** (\_Bit\_reference \_\_x, \_Bit\_reference \_\_y) noexcept
- void **std::swap** (\_Bit\_reference \_\_x, bool &\_\_y) noexcept
- void **std::swap** (bool &\_\_x, \_Bit\_reference \_\_y) noexcept

## 6.580.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 6.581 stl\_construct.h File Reference

## Namespaces

- [std](#)

## Functions

- template<typename \_T1, typename... \_Args>  
void **std::\_Construct** (\_T1 \* \_\_p, \_Args &&... \_\_args)
- template<typename \_T1 >  
void **std::\_Construct\_novalue** (\_T1 \* \_\_p)
- template<typename \_Tp >  
void **std::\_Destroy** (\_Tp \* \_\_pointer)
- template<typename \_ForwardIterator >  
void **std::\_Destroy** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_ForwardIterator, typename \_Allocator >  
void **std::\_Destroy** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Allocator & \_\_alloc)
- template<typename \_ForwardIterator, typename \_Tp >  
void **std::\_Destroy** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, allocator< \_Tp > &)
- template<typename \_ForwardIterator, typename \_Size >  
\_ForwardIterator **std::\_Destroy\_n** (\_ForwardIterator \_\_first, \_Size \_\_count)

## 6.581.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

6.582 `std_deque.h` File Reference

## Classes

- class `std::_Deque_base<_Tp, _Alloc>`
- struct `std::_Deque_iterator<_Tp, _Ref, _Ptr>`
- class `std::deque<_Tp, _Alloc>`

## Namespaces

- `std`

## Macros

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

## Functions

- constexpr `size_t std::__deque_buf_size (size_t __size)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::copy (_Deque_iterator<_Tp, _Tp &, _Tp*> __first, _Deque_iterator<_Tp, _Tp &, _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::copy (_Deque_iterator<_Tp, const _Tp &, const _Tp*> __first, _Deque_iterator<_Tp, const _Tp &, const _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::copy_backward (_Deque_iterator<_Tp, _Tp &, _Tp*> __first, _Deque_iterator<_Tp, _Tp &, _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::copy_backward (_Deque_iterator<_Tp, const _Tp &, const _Tp*> __first, _Deque_iterator<_Tp, const _Tp &, const _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`void std::fill (const _Deque_iterator<_Tp, _Tp &, _Tp*> &__first, const _Deque_iterator<_Tp, _Tp &, _Tp*> &__last, const _Tp &__value)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::move (_Deque_iterator<_Tp, _Tp &, _Tp*> __first, _Deque_iterator<_Tp, _Tp &, _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::move (_Deque_iterator<_Tp, const _Tp &, const _Tp*> __first, _Deque_iterator<_Tp, const _Tp &, const _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::move_backward (_Deque_iterator<_Tp, _Tp &, _Tp*> __first, _Deque_iterator<_Tp, _Tp &, _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`
- template<typename `_Tp`>  
`_Deque_iterator<_Tp, _Tp &, _Tp*> std::move_backward (_Deque_iterator<_Tp, const _Tp &, const _Tp*> __first, _Deque_iterator<_Tp, const _Tp &, const _Tp*> __last, _Deque_iterator<_Tp, _Tp &, _Tp*> __result)`



Generated by Doxygen

- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)`

### 6.582.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

### 6.582.2 Macro Definition Documentation

#### 6.582.2.1 `_GLIBCXX_DEQUE_BUF_SIZE`

```
#define _GLIBCXX_DEQUE_BUF_SIZE
```

This function controls the size of memory nodes.

#### Parameters

<code>__size</code>	The size of an element.
---------------------	-------------------------

#### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 88 of file `stl_deque.h`.

## 6.583 `stl_function.h` File Reference

### Classes

- struct `std::binary_function< _Arg1, _Arg2, _Result >`
- class `std::binary_negate< _Predicate >`
- class `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`

- class `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`
- class `std::const_mem_fun_ref_t< _Ret, _Tp >`
- class `std::const_mem_fun_t< _Ret, _Tp >`
- struct `std::divides< _Tp >`
- struct `std::divides< _Tp >`
- struct `std::divides< void >`
- struct `std::equal_to< _Tp >`
- struct `std::equal_to< _Tp >`
- struct `std::equal_to< void >`
- struct `std::greater< _Tp >`
- struct `std::greater< _Tp >`
- struct `std::greater< void >`
- struct `std::greater_equal< _Tp >`
- struct `std::greater_equal< _Tp >`
- struct `std::greater_equal< void >`
- struct `std::less< _Tp >`
- struct `std::less< _Tp >`
- struct `std::less< void >`
- struct `std::less_equal< _Tp >`
- struct `std::less_equal< _Tp >`
- struct `std::less_equal< void >`
- struct `std::logical_and< _Tp >`
- struct `std::logical_and< _Tp >`
- struct `std::logical_and< void >`
- struct `std::logical_not< _Tp >`
- struct `std::logical_not< _Tp >`
- struct `std::logical_not< void >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< void >`
- class `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`
- class `std::mem_fun1_t< _Ret, _Tp, _Arg >`
- class `std::mem_fun_ref_t< _Ret, _Tp >`
- class `std::mem_fun_t< _Ret, _Tp >`
- struct `std::minus< _Tp >`
- struct `std::minus< _Tp >`
- struct `std::minus< void >`
- struct `std::modulus< _Tp >`
- struct `std::modulus< _Tp >`
- struct `std::modulus< void >`
- struct `std::multiplies< _Tp >`
- struct `std::multiplies< _Tp >`
- struct `std::multiplies< void >`
- struct `std::negate< _Tp >`
- struct `std::negate< _Tp >`
- struct `std::negate< void >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< void >`
- struct `std::plus< _Tp >`
- struct `std::plus< _Tp >`

- class [std::pointer\\_to\\_binary\\_function< \\_Arg1, \\_Arg2, \\_Result >](#)
- class [std::pointer\\_to\\_unary\\_function< \\_Arg, \\_Result >](#)
- struct [std::unary\\_function< \\_Arg, \\_Result >](#)
- class [std::unary\\_negate< \\_Predicate >](#)

## Namespaces

- [std](#)

## Macros

- `#define \_\_cpp\_lib\_transparent\_operators`

## Functions

- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp > std::mem\_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`  
`const_mem_fun_t< _Ret, _Tp > std::mem\_fun (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > std::mem\_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const_mem_fun1_t< _Ret, _Tp, _Arg > std::mem\_fun (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp > std::mem\_fun\_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`  
`const_mem_fun_ref_t< _Ret, _Tp > std::mem\_fun\_ref (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem\_fun\_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem\_fun\_ref (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Predicate >`  
`\_GLIBCXX14\_CONSTEXPR unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`\_GLIBCXX14\_CONSTEXPR binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > std::ptr\_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr\_fun (_Result(*__x)(_Arg1, _Arg2))`

### 6.583.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 6.584 std\_heap.h File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void std::adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __↵`  
`value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool std::is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool std::is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance std::is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator`  
`__result, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void std::push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __↵`  
`__value, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, __↵`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

### 6.584.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 6.585 `std_iterator.h` File Reference

### Classes

- class [std::back\\_insert\\_iterator](#)< \_Container >
- class [std::front\\_insert\\_iterator](#)< \_Container >
- class [std::insert\\_iterator](#)< \_Container >
- class [std::move\\_iterator](#)< \_Iterator >
- class [std::reverse\\_iterator](#)< \_Iterator >

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define \_\_cpp\_lib\_make\_reverse\_iterator`
- `#define GLIBCXX\_MAKE\_MOVE\_IF\_NOEXCEPT\_ITERATOR(_Iter)`
- `#define GLIBCXX\_MAKE\_MOVE\_ITERATOR(_Iter)`

### Functions

- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`  
`_GLIBCXX17_CONSTEXPR _ReturnType std::\_\_make\_move\_if\_noexcept\_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = typename conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>::type>`  
`_GLIBCXX17_CONSTEXPR _ReturnType std::\_\_make\_move\_if\_noexcept\_iterator (_Tp * __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Iterator > std::\_\_make\_reverse\_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`auto std::\_\_miter\_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__miter_base(__it.base())))`

- `template<typename _Iterator >`  
`auto std::__miter_base (move_iterator< _Iterator > __it) -> decltype(__miter_base(__it.base()))`
- `template<typename _Iterator >`  
`auto std::__niter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__niter_base(__it.base())))`
- `template<typename _Iterator, typename _Container >`  
`_Iterator std::__niter_base (__gnu_cxx::__normal_iterator< _Iterator, _Container > __it)`
- `template<typename _Iterator >`  
`auto std::__niter_base (move_iterator< _Iterator > __it) -> decltype(make_move_iterator(__niter_base(__it.base())))`
- `template<typename _Container >`  
`back_insert_iterator< _Container > std::back_inserter (_Container & __x)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > std::front_inserter (_Container & __x)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > std::inserter (_Container & __x, _Iterator __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR move_iterator< _Iterator > std::make_move_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const reverse_iterator< _Iterator > & __x, const reverse_iterator< _Iterator > & __y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const reverse_iterator< _IteratorL > & __x, const reverse_iterator< _IteratorR > & __y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator!= (const __normal_iterator< _IteratorL, _Container > & __lhs, const __normal_iterator< _IteratorR, _Container > & __rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator!= (const __normal_iterator< _Iterator, _Container > & __lhs, const __normal_iterator< _Iterator, _Container > & __rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const move_iterator< _IteratorL > & __x, const move_iterator< _IteratorR > & __y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator!= (const move_iterator< _Iterator > & __x, const move_iterator< _Iterator > & __y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > & __x)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > __gnu_cxx::operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > & __i) noexcept`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > & __x)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR auto std::operator- (const reverse_iterator< _IteratorL > & __x, const reverse_iterator< _IteratorR > & __y) -> decltype(__y.base() - __x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`auto __gnu_cxx::operator- (const __normal_iterator< _IteratorL, _Container > & __lhs, const __normal_iterator< _IteratorR, _Container > & __rhs) noexcept -> decltype(__lhs.base() - __rhs.base())`

- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container >::difference_type __gnu_cxx::operator-` `(const __normal_iterator<`  
`_Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR auto std::operator-` `(const move_iterator< _IteratorL > &__x, const move_↵`  
`iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<` `(const reverse_iterator< _Iterator > &__x, const reverse_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<` `(const reverse_iterator< _IteratorL > &__x, const reverse_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator<` `(const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator<` `(const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_↵`  
`iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<` `(const move_iterator< _IteratorL > &__x, const move_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<` `(const move_iterator< _Iterator > &__x, const move_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<=` `(const reverse_iterator< _Iterator > &__x, const reverse_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<=` `(const reverse_iterator< _IteratorL > &__x, const reverse_↵`  
`_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator<=` `(const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator<=` `(const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_↵`  
`iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<=` `(const move_iterator< _IteratorL > &__x, const move_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator<=` `(const move_iterator< _Iterator > &__x, const move_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator==` `(const reverse_iterator< _Iterator > &__x, const reverse_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator==` `(const reverse_iterator< _IteratorL > &__x, const reverse_↵`  
`_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator==` `(const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator==` `(const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_↵`  
`iterator< _Iterator, _Container > &__rhs) noexcept`



- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator== (const move_iterator< _IteratorL > &__x, const move_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator== (const move_iterator< _Iterator > &__x, const move_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_↵`  
`iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const move_iterator< _IteratorL > &__x, const move_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator> (const move_iterator< _Iterator > &__x, const move_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_↵`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`bool gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_↵`  
`iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_↵`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`_GLIBCXX17_CONSTEXPR bool std::operator>= (const move_iterator< _Iterator > &__x, const move_↵`  
`iterator< _Iterator > &__y)`

#### 6.585.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

6.586 `stl_iterator.h` File Reference

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Functions

- `template<typename _Iterator >`  
`auto __gnu_debug::__base (const std::reverse\_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_iterator(__base(__it.base())))`
- `template<typename _Iterator >`  
`auto __gnu_debug::__base (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__base(__it.base())))`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::reverse\_iterator< _Iterator > &__first, const std::reverse\_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::move\_iterator< _Iterator > &__first, const std::move\_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`auto __gnu_debug::__unsafe (const std::reverse\_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator >`  
`auto __gnu_debug::__unsafe (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator >`  
`bool __gnu_debug::__valid_range (const std::reverse\_iterator< _Iterator > &__first, const std::reverse\_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator >`  
`bool __gnu_debug::__valid_range (const std::move\_iterator< _Iterator > &__first, const std::move\_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`

## 6.586.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.587 `stl_iterator_base_funcs.h` File Reference

## Classes

- struct [std::\\_List\\_const\\_iterator](#)< \_Tp >
- struct [std::\\_List\\_iterator](#)< \_Tp >

## Namespaces

- [std](#)

## Functions

- `template<typename _InputIterator, typename _Distance >`  
`_GLIBCXX14_CONSTEXPR void std::advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`_GLIBCXX14_CONSTEXPR void std::advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_GLIBCXX14_CONSTEXPR void std::advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _InputIterator >`  
`_GLIBCXX14_CONSTEXPR iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`_GLIBCXX14_CONSTEXPR iterator_traits< _RandomAccessIterator >::difference_type std::distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Distance >`  
`_GLIBCXX17_CONSTEXPR void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator >`  
`_GLIBCXX17_CONSTEXPR iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator >`  
`_GLIBCXX17_CONSTEXPR _InputIterator std::next (_InputIterator __x, typename iterator_traits< _InputIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`  
`_GLIBCXX17_CONSTEXPR _BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)`

### 6.587.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

## 6.588 `std::iterator_base_types.h` File Reference

### Classes

- `struct std::iterator_traits< _Iterator, typename >`
- `struct std::bidirectional_iterator_tag`
- `struct std::forward_iterator_tag`
- `struct std::input_iterator_tag`
- `struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
- `struct std::iterator_traits< _Tp * >`
- `struct std::iterator_traits< const _Tp * >`
- `struct std::output_iterator_tag`
- `struct std::random_access_iterator_tag`

## Namespaces

- [std](#)

## Typedefs

- `template<typename _InIter >`  
using **`std::_RequireInputIter`** = `typename enable_if< is_convertible< typename iterator_traits< _InIter >::iterator_category, input_iterator_tag >::value >::type`

## Functions

- `template<typename _Iter >`  
`constexpr iterator_traits< _Iter >::iterator_category std::\_\_iterator\_category (const _Iter &)`

### 6.588.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

## 6.589 `std_list.h` File Reference

### Classes

- struct [std::\\_\\_detail::\\_List\\_node\\_base](#)
- struct [std::\\_\\_detail::\\_List\\_node\\_header](#)
- class [std::\\_List\\_base< \\_Tp, \\_Alloc >](#)
- struct [std::\\_List\\_const\\_iterator< \\_Tp >](#)
- struct [std::\\_List\\_iterator< \\_Tp >](#)
- struct [std::\\_List\\_node< \\_Tp >](#)
- class [std::list< \\_Tp, \\_Alloc >](#)

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<typename _Val >`  
`bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val >`  
`bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`_GLIBCXX_END_NAMESPACE_CXX11 bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)`

### 6.589.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 6.590 `stl_map.h` File Reference

### Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`

### Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`  
`noexcept(/*conditional */)`

## 6.590.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

6.591 `std_multimap.h` File Reference

## Classes

- class `std::map< \_Key, \_Tp, \_Compare, \_Alloc >`
- class `std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >`

## Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

### 6.591.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

## 6.592 stl\_multiset.h File Reference

### Classes

- class `std::multiset< _Key, _Compare, _Alloc >`
- class `std::set< _Key, _Compare, _Alloc >`

### Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

## 6.592.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 6.593 std\_numeric.h File Reference

## Namespaces

- `std`

## Functions

- `template<typename _InputIterator, typename _Tp >`  
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`  
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`_Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`  
`_Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

### 6.593.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

## 6.594 std\_pair.h File Reference

### Classes

- struct `std::pair<_T1, _T2 >`
- struct `std::piecewise_construct_t`
- class `std::tuple<_Elements >`

### Namespaces

- `std`

### Functions

- `template<typename _T1, typename _T2 >`  
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1, typename _T2 >`  
`enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2 >`  
`enable_if<! __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type std::swap (pair< _T1, _T2 > &, pair< _T1, _T2 > &)=delete`

## Variables

- `_GLIBCXX17_INLINE` constexpr `piecewise_construct_t` [std::piecewise\\_construct](#)

## 6.594.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

6.595 `std::queue.h` File Reference

## Classes

- class [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare>](#)
- class [std::queue<\\_Tp, \\_Sequence>](#)

## Namespaces

- [std](#)

## Functions

- `template<typename _Tp, typename _Seq>`  
`bool std::operator!= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator< (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator<= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator== (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator> (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator>= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`enable_if<__is_swappable<_Seq>::value>::type std::swap (queue<_Tp, _Seq> &__x, queue<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Sequence, typename _Compare>`  
`enable_if<__and<__is_swappable<_Sequence>, __is_swappable<_Compare>>::value>::type std::swap (priority_queue<_Tp, _Sequence, _Compare> &__x, priority_queue<_Tp, _Sequence, _Compare> &__y) noexcept(noexcept(__x.swap(__y)))`

## 6.595.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 6.596 `std_raw_storage_iter.h` File Reference

### Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp>`

### Namespaces

- `std`

#### 6.596.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.597 `std_relops.h` File Reference

### Namespaces

- `std`
- `std::rel_ops`

### Functions

- `template<class _Tp>`  
`bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`  
`bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)`

#### 6.597.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std::utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous\\_overloads](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads)

Short summary: the `rel_ops` operators should be avoided for the present.

6.598 `std_set.h` File Reference

## Classes

- class `std::multiset<_Key, _Compare, _Alloc>`
- class `std::set<_Key, _Compare, _Alloc>`

## Namespaces

- `std`

## Functions

- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
bool `std::operator!=` (const set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, const set< `_Key`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
bool `std::operator<` (const set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, const set< `_Key`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
bool `std::operator<=` (const set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, const set< `_Key`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
bool `std::operator==` (const set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, const set< `_Key`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
bool `std::operator>` (const set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, const set< `_Key`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
bool `std::operator>=` (const set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, const set< `_Key`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Key`, typename `_Compare`, typename `_Alloc`>  
void `std::swap` (set< `_Key`, `_Compare`, `_Alloc` > &\_\_x, set< `_Key`, `_Compare`, `_Alloc` > &\_\_y) noexcept(*/\*conditional \*/*)

## 6.598.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

6.599 `std_stack.h` File Reference

## Classes

- class `std::stack<_Tp, _Sequence>`

## Namespaces

- `std`

## Functions

- `template<typename _Tp, typename _Seq >`  
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`enable_if< __is_swappable< _Seq >::value >::type std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

### 6.599.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

## 6.600 stl\_tempbuf.h File Reference

### Classes

- class [std::\\_Temporary\\_buffer<\\_ForwardIterator, \\_Tp >](#)

### Namespaces

- [std](#)

### Functions

- `template<typename _Pointer, typename _ForwardIterator >`  
`void std::\_\_uninitialized\_construct\_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _Tp >`  
`pair< _Tp *, ptrdiff_t > std::get\_temporary\_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp >`  
`void std::return\_temporary\_buffer (_Tp *__p)`

### 6.600.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

6.601 `std_tree.h` File Reference

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_generic_associative_lookup`

## Enumerations

- `enum _Rb_tree_color { _S_red, _S_black }`

## Functions

- `unsigned int std::Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()`
- `_Rb_tree_node_base * std::Rb_tree_decrement ( _Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * std::Rb_tree_decrement (const _Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * std::Rb_tree_increment ( _Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * std::Rb_tree_increment (const _Rb_tree_node_base *__x) throw ()`
- `void std::Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * std::Rb_tree_rebalance_for_erase ( _Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `template<typename _Val >  
bool std::operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)  
noexcept`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Val >  
bool std::operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)  
noexcept`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool std::operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
void std::swap ( _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`

### 6.601.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

## 6.602 `std_uninitialized.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::uninitialized_copy_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator & __alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`pair< _InputIterator, _ForwardIterator > std::uninitialized_copy_n_pair (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`pair< _RandomAccessIterator, _ForwardIterator > std::uninitialized_copy_n_pair (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`pair< _InputIterator, _ForwardIterator > std::uninitialized_copy_n_pair (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator >`  
`void std::uninitialized_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void std::uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`  
`_ForwardIterator std::uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`  
`_ForwardIterator std::uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _Allocator & __alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`_ForwardIterator std::uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocator< _Tp > &)`

- `template<typename _ForwardIterator >`  
`void std::__uninitialized_default_noval (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size >`  
`_ForwardIterator std::__uninitialized_default_noval_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`  
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator<_Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`  
`_ForwardIterator std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator<_Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void std::__uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`_ForwardIterator std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`

### 6.602.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.603 `std_vector.h` File Reference

### Classes

- struct `std::_Vector_base<_Tp, _Alloc>`
- class `std::vector<_Tp, _Alloc>`



## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_ASAN_ANNOTATE_BEFORE_DEALLOC`
- `#define _GLIBCXX_ASAN_ANNOTATE_GREW(n)`
- `#define _GLIBCXX_ASAN_ANNOTATE_GROW(n)`
- `#define _GLIBCXX_ASAN_ANNOTATE_REINIT`
- `#define _GLIBCXX_ASAN_ANNOTATE_SHRINK(n)`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)`

### 6.603.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 6.604 `stream_iterator.h` File Reference

### Classes

- class [std::istream\\_iterator< \\_Tp, \\_CharT, \\_Traits, \\_Dist >](#)
- class [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >](#)

## Namespaces

- [std](#)

## Functions

- `template<class _Tp, class _CharT, class _Traits, class _Dist >`  
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`  
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`

### 6.604.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.605 streambuf File Reference

### Classes

- class [std::basic\\_streambuf< \\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

### Macros

- `#define \_GLIBXX\_STREAMBUF`
- `#define \_IsUnused`

## Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::copy\_streambufs\_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`  
`streamsize std::copy\_streambufs\_eof (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sout, bool &__ineof)`
- `template<>`  
`streamsize std::copy\_streambufs\_eof (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sout, bool &__ineof)`

### 6.605.1 Detailed Description

This is a Standard C++ Library header.

## 6.606 streambuf.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _STREAMBUF_TCC`

### Functions

- `template<typename _CharT, typename _Traits >  
streamsize std::__copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >  
streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`

### 6.606.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<streambuf>`.

## 6.607 streambuf\_iterator.h File Reference

### Classes

- class [std::istreambuf\\_iterator< \\_CharT, \\_Traits >](#)
- class [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

## Functions

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`  
`copy_move_a2 ( _CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`  
`copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _CharT, typename _Distance >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type std::advance (istreambuf_iterator<`  
`_CharT > &__i, _Distance __n)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`  
`__result)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`  
`_Traits > &__b)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`  
`_Traits > &__b)`

## 6.607.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 6.608 string File Reference

## Macros

- `#define _GLIBCXX_STRING`

## 6.608.1 Detailed Description

This is a Standard C++ Library header.

## 6.609 string File Reference

## Classes

- class `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_STRING`

## Typedefs

- `typedef basic_string< char > __gnu_debug::string`
- `typedef basic_string< wchar_t > __gnu_debug::wstring`

## Functions

- `template<typename _CharT, typename _Traits, typename _Allocator >  
std::basic_istream< _CharT, _Traits > & __gnu_debug::getline (std::basic_istream< _CharT, _Traits > &__is,  
basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
std::basic_istream< _CharT, _Traits > & __gnu_debug::getline (std::basic_istream< _CharT, _Traits > &__is,  
basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<  
_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, ↵  
_Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const _CharT *__lhs, const basic_↵  
string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (_CharT __lhs, const basic_string< ↵  
_CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, ↵  
_Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, ↵  
_Allocator > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<  
_CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_debug::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > & __gnu_debug::operator>> (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

### 6.609.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.610 string File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_STRING`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Up >`  
`void std::experimental::fundamentals_v2::erase (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (basic_string< _CharT, _Traits, _Alloc > &__cont, _↵ Predicate __pred)`

### 6.610.1 Detailed Description

This is a TS C++ Library header.

## 6.611 string\_conversions.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`  
`_Ret __gnu_cxx::__stoa (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str, std::size_t *__idx, _Base... __base)`
- `template<typename _String, typename _CharT = typename _String::value_type>`  
`_String __gnu_cxx::__to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std↵ ::size_t __n, const _CharT *__fmt,...)`

## 6.611.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.612 string\_view File Reference

## Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view<\\_CharT, \\_Traits>](#)
- struct [std::hash<\\_Tp>](#)

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_experimental_string_view`
- `#define _GLIBCXX_EXPERIMENTAL_STRING_VIEW`

## Typedefs

- `template<typename _Tp>`  
using **std::experimental::fundamentals\_v1::\_\_detail::\_\_idt** = `common_type_t<_Tp>`
- using **std::experimental::fundamentals\_v1::string\_view** = `basic_string_view<char>`
- using **std::experimental::fundamentals\_v1::u16string\_view** = `basic_string_view<char16_t>`
- using **std::experimental::fundamentals\_v1::u32string\_view** = `basic_string_view<char32_t>`
- using **std::experimental::fundamentals\_v1::wstring\_view** = `basic_string_view<wchar_t>`

## Functions

- `template<typename _CharT, typename _Traits>`  
constexpr bool **std::experimental::fundamentals\_v1::operator!=** (`basic_string_view<_CharT, _Traits> __x`, `basic_string_view<_CharT, _Traits> __y`) noexcept
- `template<typename _CharT, typename _Traits>`  
constexpr bool **std::experimental::fundamentals\_v1::operator!=** (`basic_string_view<_CharT, _Traits> __x`, `__detail::__idt<basic_string_view<_CharT, _Traits>> __y`) noexcept
- `template<typename _CharT, typename _Traits>`  
constexpr bool **std::experimental::fundamentals\_v1::operator!=** (`__detail::__idt<basic_string_view<_CharT, _Traits>> __x`, `basic_string_view<_CharT, _Traits> __y`) noexcept
- constexpr `basic_string_view<char>` **std::experimental::literals::string\_view\_literals::operator""sv** (`const char *__str`, `size_t __len`) noexcept
- constexpr `basic_string_view<wchar_t>` **std::experimental::literals::string\_view\_literals::operator""sv** (`const wchar_t *__str`, `size_t __len`) noexcept
- constexpr `basic_string_view<char16_t>` **std::experimental::literals::string\_view\_literals::operator""sv** (`const char16_t *__str`, `size_t __len`) noexcept



- constexpr basic\_string\_view< char32\_t > **std::experimental::literals::string\_view\_literals::operator""sv** (const char32\_t \*\_\_str, size\_t \_\_len) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**< (basic\_string\_view< \_CharT, \_Traits > \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**< (basic\_string\_view< \_CharT, \_Traits > \_\_x, \_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**< (\_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
basic\_ostream< \_CharT, \_Traits > & **std::experimental::fundamentals\_v1::operator**<< (basic\_ostream< \_CharT, \_Traits > &\_\_os, basic\_string\_view< \_CharT, \_Traits > \_\_str)
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**<= (basic\_string\_view< \_CharT, \_Traits > \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**<= (basic\_string\_view< \_CharT, \_Traits > \_\_x, \_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**<= (\_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**== (basic\_string\_view< \_CharT, \_Traits > \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**== (basic\_string\_view< \_CharT, \_Traits > \_\_x, \_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**== (\_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**> (basic\_string\_view< \_CharT, \_Traits > \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**> (basic\_string\_view< \_CharT, \_Traits > \_\_x, \_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**> (\_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**>= (basic\_string\_view< \_CharT, \_Traits > \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**>= (basic\_string\_view< \_CharT, \_Traits > \_\_x, \_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_y) noexcept
- template<typename \_CharT, typename \_Traits >  
constexpr bool **std::experimental::fundamentals\_v1::operator**>= (\_\_detail::\_\_idt< basic\_string\_view< \_CharT, \_Traits >> \_\_x, basic\_string\_view< \_CharT, \_Traits > \_\_y) noexcept

### 6.612.1 Detailed Description

This is a TS C++ Library header.

## 6.613 string\_view.tcc File Reference

### Macros

- `#define _GLIBCXX_STRING_VIEW_TCC`

#### 6.613.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string_view>`.

## 6.614 string\_view.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_STRING_VIEW_TCC`

#### 6.614.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/string_view>`.

## 6.615 stringfwd.h File Reference

### Classes

- class [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>](#)
- struct [std::char\\_traits<\\_CharT>](#)

### Namespaces

- [std](#)

### Typedefs

- typedef [basic\\_string<char>](#) [std::string](#)
- typedef [basic\\_string<char16\\_t>](#) [std::u16string](#)
- typedef [basic\\_string<char32\\_t>](#) [std::u32string](#)
- typedef [basic\\_string<wchar\\_t>](#) [std::wstring](#)

#### 6.615.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

### 6.616 `strstream` File Reference

#### Namespaces

- [std](#)

#### 6.616.1 Detailed Description

This is a Standard C++ Library header.

### 6.617 `synth_access_traits.hpp` File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC`
- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC`

#### 6.617.1 Detailed Description

Contains an implementation class for a patricia tree.

### 6.618 `system_error` File Reference

#### Classes

- class [std::\\_V2::error\\_category](#)
- struct [std::error\\_code](#)
- struct [std::error\\_condition](#)
- struct [std::hash< \\_Tp >](#)
- struct [std::hash< error\\_code >](#)
- struct [std::is\\_error\\_code\\_enum< \\_Tp >](#)
- struct [std::is\\_error\\_condition\\_enum< \\_Tp >](#)
- class [std::system\\_error](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_SYSTEM_ERROR`

## Functions

- `const error_category & std::_V2::generic_category () noexcept`
- `error_code std::make_error_code (errc __e) noexcept`
- `error_condition std::make_error_condition (errc __e) noexcept`
- `bool std::operator!= (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator!= (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `bool std::operator!= (const error_condition &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator!= (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `bool std::operator< (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator< (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)`
- `bool std::operator== (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator== (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `bool std::operator== (const error_condition &__lhs, const error_code &__rhs) noexcept`
- `bool std::operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `const error_category & std::_V2::system_category () noexcept`

## Variables

- `error_code std::make_error_code (errc) noexcept`
- `error_condition std::make_error_condition (errc) noexcept`

## 6.618.1 Detailed Description

This is a Standard C++ Library header.

## 6.619 system\_error File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_SYSTEM_ERROR`

## Variables

- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_error_code_enum_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_error_condition_enum_v`

## 6.619.1 Detailed Description

This is a TS C++ Library header.

## 6.620 tag\_and\_trait.hpp File Reference

### Classes

- `struct \_\_gnu\_pbds::associative\_tag`
- `struct \_\_gnu\_pbds::basic\_branch\_tag`
- `struct \_\_gnu\_pbds::basic\_hash\_tag`
- `struct \_\_gnu\_pbds::basic\_invalidation\_guarantee`
- `struct \_\_gnu\_pbds::binary\_heap\_tag`
- `struct \_\_gnu\_pbds::binomial\_heap\_tag`
- `struct \_\_gnu\_pbds::cc\_hash\_tag`
- `struct \_\_gnu\_pbds::container\_tag`
- `struct \_\_gnu\_pbds::container\_traits< Cntnr >`
- `struct \_\_gnu\_pbds::container\_traits\_base< \_Tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< binary\_heap\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< binomial\_heap\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< cc\_hash\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< gp\_hash\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< list\_update\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< ov\_tree\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< pairing\_heap\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< pat\_trie\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< rb\_tree\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< rc\_binomial\_heap\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< splay\_tree\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< thin\_heap\_tag >`
- `struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_Tl >`
- `struct \_\_gnu\_pbds::gp\_hash\_tag`
- `struct \_\_gnu\_pbds::list\_update\_tag`
- `struct \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_Tp4 >`
- `struct \_\_gnu\_pbds::null\_type`
- `struct \_\_gnu\_pbds::ov\_tree\_tag`
- `struct \_\_gnu\_pbds::pairing\_heap\_tag`
- `struct \_\_gnu\_pbds::pat\_trie\_tag`
- `struct \_\_gnu\_pbds::point\_invalidation\_guarantee`
- `struct \_\_gnu\_pbds::priority\_queue\_tag`
- `struct \_\_gnu\_pbds::range\_invalidation\_guarantee`

- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Typedefs

- typedef void [\\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

#### 6.620.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

## 6.621 tags.h File Reference

#### Classes

- struct [\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::balanced\\_tag](#)
- struct [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#)
- struct [\\_\\_gnu\\_parallel::default\\_parallel\\_tag](#)
- struct [\\_\\_gnu\\_parallel::equal\\_split\\_tag](#)
- struct [\\_\\_gnu\\_parallel::exact\\_tag](#)
- struct [\\_\\_gnu\\_parallel::find\\_tag](#)
- struct [\\_\\_gnu\\_parallel::growing\\_blocks\\_tag](#)
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag](#)
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#)
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag](#)
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_tag](#)
- struct [\\_\\_gnu\\_parallel::parallel\\_tag](#)
- struct [\\_\\_gnu\\_parallel::quicksort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::sampling\\_tag](#)
- struct [\\_\\_gnu\\_parallel::sequential\\_tag](#)
- struct [\\_\\_gnu\\_parallel::unbalanced\\_tag](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

### 6.621.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

## 6.622 tgmath.h File Reference

## Macros

- `#define _GLIBCXX_TGMATH_H`

### 6.622.1 Detailed Description

This is a Standard C++ Library header.

## 6.623 thin\_heap.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## Enumerations

- enum { `num_distinct_rank_bounds` }

## Variables

- static const std::size\_t **\_\_gnu\_pbds::detail::g\_a\_rank\_bounds** [num\_distinct\_rank\_bounds]

## 6.623.1 Detailed Description

Contains an implementation class for a thin heap.

## 6.624 thread File Reference

## Classes

- struct [std::hash< thread::id >](#)
- class [std::thread](#)
- class [std::thread::id](#)

## Namespaces

- [std](#)
- [std::this\\_thread](#)

## Macros

- `#define _GLIBCXX_THREAD`

## Functions

- void **std::this\_thread::\_\_sleep\_for** (chrono::seconds, chrono::nanoseconds)
- thread::id [std::this\\_thread::get\\_id](#) () noexcept
- bool **std::operator!=** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator<** (thread::id \_\_x, thread::id \_\_y) noexcept
- template<class \_CharT, class \_Traits >  
basic\_ostream< \_CharT, \_Traits > & **std::operator<<** (basic\_ostream< \_CharT, \_Traits > &\_\_out, thread::id \_\_id)
- bool **std::operator<=** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator==** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator>** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator>=** (thread::id \_\_x, thread::id \_\_y) noexcept
- template<typename \_Rep, typename \_Period >  
void [std::this\\_thread::sleep\\_for](#) (const chrono::duration< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
void [std::this\\_thread::sleep\\_until](#) (const chrono::time\_point< \_Clock, \_Duration > &\_\_atime)
- void **std::swap** (thread &\_\_x, thread &\_\_y) noexcept
- void [std::this\\_thread::yield](#) () noexcept



### 6.624.1 Detailed Description

This is a Standard C++ Library header.

## 6.625 `throw_allocator.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::annotate\\_base](#)
- struct [\\_\\_gnu\\_cxx::condition\\_base](#)
- struct [\\_\\_gnu\\_cxx::forced\\_error](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::limit\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::group\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::never\\_adjustor](#)
- class [\\_\\_gnu\\_cxx::throw\\_allocator\\_base< \\_Tp, \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_limit< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_random< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_base< \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_throw\\_forced\\_error\(\)](#)
- [template<typename \\_Tp, typename \\_Cond >](#)  
[bool \\_\\_gnu\\_cxx::operator!= \(const throw\\_allocator\\_base< \\_Tp, \\_Cond > &, const throw\\_allocator\\_base< \\_Tp, \\_Cond > &\)](#)
- [template<typename \\_Cond >](#)  
[throw\\_value\\_base< \\_Cond > \\_\\_gnu\\_cxx::operator\\* \(const throw\\_value\\_base< \\_Cond > &\\_\\_a, const throw\\_value\\_base< \\_Cond > &\\_\\_b\)](#)
- [template<typename \\_Cond >](#)  
[throw\\_value\\_base< \\_Cond > \\_\\_gnu\\_cxx::operator+ \(const throw\\_value\\_base< \\_Cond > &\\_\\_a, const throw\\_value\\_base< \\_Cond > &\\_\\_b\)](#)
- [template<typename \\_Cond >](#)  
[throw\\_value\\_base< \\_Cond > \\_\\_gnu\\_cxx::operator- \(const throw\\_value\\_base< \\_Cond > &\\_\\_a, const throw\\_value\\_base< \\_Cond > &\\_\\_b\)](#)

- `template<typename _Cond >`  
`bool __gnu_cxx::operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `std::ostream & __gnu_cxx::operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Cond >`  
`bool __gnu_cxx::operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp, typename _Cond >`  
`bool __gnu_cxx::operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond >`  
`void __gnu_cxx::swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`

#### 6.625.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

## 6.626 time\_members.h File Reference

### Namespaces

- [std](#)

#### 6.626.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 6.627 trace\_fn\_imps.hpp File Reference

#### 6.627.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 6.628 trace\_fn\_imps.hpp File Reference

#### 6.628.1 Detailed Description

Contains implementations of `cc_ht_map_'s` trace-mode functions.

## 6.629 `trace_fn_imps.hpp` File Reference

### 6.629.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

## 6.630 `trace_fn_imps.hpp` File Reference

### 6.630.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.631 `trace_fn_imps.hpp` File Reference

### 6.631.1 Detailed Description

Contains implementations of `lu_map_`.

## 6.632 `trace_fn_imps.hpp` File Reference

### 6.632.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 6.633 `trace_fn_imps.hpp` File Reference

### 6.633.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 6.634 `trace_fn_imps.hpp` File Reference

### 6.634.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 6.635 traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.635.1 Detailed Description

Contains an implementation for bin\_search\_tree\_.

## 6.636 traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Data, Cmp\\_Fn, Node\\_Update, Tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Data, \\_ATraits, Node\\_Update, Tag, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

#### 6.636.1 Detailed Description

Contains an implementation class for tree-like classes.

## 6.637 traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.637.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 6.638 traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Mapped, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, null\\_type, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.638.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 6.639 traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 6.639.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 6.640 traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 6.640.1 Detailed Description

Contains an implementation for splay\_tree\_.

## 6.641 tree\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::tree\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BRANCH_POLICY_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.641.1 Detailed Description

Contains tree-related policies.

## 6.642 tree\_trace\_base.hpp File Reference

#### 6.642.1 Detailed Description

Contains tree-related policies.

## 6.643 trie\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- struct [\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits](#)< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_TRIE_POLICY_BASE`

#### 6.643.1 Detailed Description

Contains trie-related policies.

## 6.644 trie\_policy\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::trie\\_policy\\_base](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.644.1 Detailed Description

Contains an implementation of `trie_policy_base`.

## 6.645 trie\_string\_access\_traits\_imp.hpp File Reference

### 6.645.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

## 6.646 tuple File Reference

### Classes

- struct [std::\\_Tuple\\_impl<\\_Idx, \\_Elements>](#)
- struct [std::\\_Tuple\\_impl<\\_Idx, \\_Head, \\_Tail...>](#)
- class [std::tuple<\\_Elements>](#)
- class [std::tuple<\\_Elements>](#)
- class [std::tuple<\\_T1, \\_T2>](#)
- struct [std::tuple\\_element<0, tuple<\\_Head, \\_Tail...>>](#)
- struct [std::tuple\\_element<\\_\\_i, tuple<\\_Head, \\_Tail...>>](#)
- struct [std::tuple\\_element<\\_\\_i, tuple<>>](#)
- struct [std::tuple\\_size<tuple<\\_Elements...>>](#)
- struct [std::uses\\_allocator<tuple<\\_Types...>, \\_Alloc>](#)

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_tuples_by_type`
- `#define _GLIBCXX_TUPLE`

### Typedefs

- `template<typename _Tp>  
using std::__empty_not_final = typename conditional<__is_final(_Tp), false_type, __is_empty_non_tuple<_Tp>>::type`



## Functions

- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & std::__get_helper ( _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & std::__get_helper (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr _Head & std::__get_helper2 ( _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr const _Head & std::__get_helper2 (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > std::forward_as_tuple ( _Elements &&... __args) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp & std::get (tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp && std::get (tuple< _Types... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp & std::get (const tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp && std::get (const tuple< _Types... > &&__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::type... > std::make_tuple ( _Elements &&... __args)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator!= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator< (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator<= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator== (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator> (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator>= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _Elements>`  
`enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`enable_if<!__and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__, tuple< _Elements... > &__)=delete`

- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > std::tie ( _Elements &... __args) noexcept`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`  
`constexpr auto std::tuple_cat ( _Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls... >::__type`

#### Variables

- `_GLIBCXX17_INLINE constexpr _Swallow_assign std::ignore`

#### 6.646.1 Detailed Description

This is a Standard C++ Library header.

## 6.647 tuple File Reference

#### Namespaces

- `std`

#### Macros

- `#define __cpp_lib_experimental_tuple`
- `#define _GLIBCXX_EXPERIMENTAL_TUPLE`

#### Functions

- `template<typename _Fn, typename _Tuple, std::size_t... _Idx>`  
`decltype(auto) constexpr std::experimental::fundamentals_v1::__apply_impl ( _Fn &&__f, _Tuple &&__t, std::index_sequence< _Idx... >)`
- `template<typename _Fn, typename _Tuple >`  
`decltype(auto) constexpr std::experimental::fundamentals_v1::apply ( _Fn &&__f, _Tuple &&__t)`

#### Variables

- `template<typename _Tp >`  
`constexpr size_t std::experimental::fundamentals_v1::tuple_size_v`

#### 6.647.1 Detailed Description

This is a TS C++ Library header.

## 6.648 type\_traits File Reference

### Classes

- struct [std::\\_\\_add\\_pointer\\_helper](#)< \_Tp, bool >
- struct [std::\\_\\_detector](#)< \_Default, \_AlwaysVoid, \_Op, \_Args >
- struct [std::\\_\\_detector](#)< \_Default, \_\_void\_t< \_Op< \_Args... > >, \_Op, \_Args... >
- struct [std::\\_\\_is\\_nullptr\\_t](#)< \_Tp >
- struct [std::\\_\\_is\\_trivially\\_copy\\_assignable\\_impl](#)< \_Tp, bool >
- struct [std::\\_\\_is\\_trivially\\_copy\\_constructible\\_impl](#)< \_Tp, bool >
- struct [std::\\_\\_is\\_trivially\\_move\\_assignable\\_impl](#)< \_Tp, bool >
- struct [std::\\_\\_is\\_trivially\\_move\\_constructible\\_impl](#)< \_Tp, bool >
- struct [std::add\\_const](#)< \_Tp >
- struct [std::add\\_cv](#)< \_Tp >
- struct [std::add\\_lvalue\\_reference](#)< \_Tp >
- struct [std::add\\_rvalue\\_reference](#)< \_Tp >
- struct [std::add\\_volatile](#)< \_Tp >
- struct [std::aligned\\_storage](#)< \_Len, \_Align >
- struct [std::aligned\\_union](#)< \_Len, \_Types >
- struct [std::alignment\\_of](#)< \_Tp >
- struct [std::common\\_type](#)< \_Tp >
- struct [std::conditional](#)< \_Cond, \_Iftrue, \_Iffalse >
- struct [std::conditional](#)< \_Cond, \_Iftrue, \_Iffalse >
- class [std::decay](#)< \_Tp >
- struct [std::enable\\_if](#)< bool, \_Tp >
- struct [std::extent](#)< typename, \_Uint >
- struct [std::extent](#)< typename, \_Uint >
- struct [std::has\\_virtual\\_destructor](#)< \_Tp >
- struct [std::integral\\_constant](#)< \_Tp, \_\_v >
- struct [std::is\\_abstract](#)< \_Tp >
- struct [std::is\\_arithmetic](#)< \_Tp >
- struct [std::is\\_array](#)< typename >
- struct [std::is\\_assignable](#)< \_Tp, \_Up >
- struct [std::is\\_base\\_of](#)< \_Base, \_Derived >
- struct [std::is\\_class](#)< \_Tp >
- struct [std::is\\_compound](#)< \_Tp >
- struct [std::is\\_const](#)< typename >
- struct [std::is\\_constructible](#)< \_Tp, \_Args >
- struct [std::is\\_convertible](#)< \_From, \_To >
- struct [std::is\\_copy\\_assignable](#)< \_Tp >
- struct [std::is\\_copy\\_constructible](#)< \_Tp >
- struct [std::is\\_default\\_constructible](#)< \_Tp >
- struct [std::is\\_destructible](#)< \_Tp >
- struct [std::is\\_empty](#)< \_Tp >
- struct [std::is\\_enum](#)< \_Tp >
- struct [std::is\\_final](#)< \_Tp >
- struct [std::is\\_floating\\_point](#)< \_Tp >
- struct [std::is\\_function](#)< typename >
- struct [std::is\\_function](#)< typename >
- struct [std::is\\_fundamental](#)< \_Tp >

- struct `std::is_integral< _Tp >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_nothrow_swappable< _Tp >`
- struct `std::is_nothrow_swappable_with< _Tp, _Up >`
- struct `std::is_null_pointer< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< typename, typename >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_swappable< _Tp >`
- struct `std::is_swappable_with< _Tp, _Up >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< typename >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< typename >`
- class `std::reference_wrapper< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`
- struct `std::remove_cv< _Tp >`
- struct `std::remove_cv< _Tp >`
- struct `std::remove_extent< _Tp >`
- struct `std::remove_pointer< _Tp >`
- struct `std::remove_reference< _Tp >`

- struct [std::remove\\_volatile< \\_Tp >](#)
- class [std::result\\_of< \\_Signature >](#)
- class [std::tuple< \\_Elements >](#)
- struct [std::underlying\\_type< \\_Tp >](#)

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_integral_constant_callable`
- `#define __cpp_lib_is_final`
- `#define __cpp_lib_is_null_pointer`
- `#define __cpp_lib_is_swappable`
- `#define __cpp_lib_result_of_sfinae`
- `#define __cpp_lib_transformation_trait_aliases`
- `#define __cpp_lib_void_t`
- `#define \_GLIBCXX\_HAS\_NESTED\_TYPE(_NTYPE)`
- `#define \_GLIBCXX\_TYPE\_TRAITS`

## Typedefs

- `template<bool __v>  
using std::\_\_bool\_constant = integral_constant< bool, __v >`
- `template<typename _Fn, typename... _Args>  
using std::\_\_call\_is\_nothrow = __call_is_nothrow< __invoke_result< _Fn, _Args... >, _Fn, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>  
using std::\_\_detected\_or = __detector< _Default, void, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>  
using std::\_\_detected\_or\_t = typename __detected_or< _Default, _Op, _Args... >::type`
- `template<bool _Cond, typename _Tp = void>  
using std::\_\_enable\_if\_t = typename enable_if< _Cond, _Tp >::type`
- `template<typename _Tp, typename... _Args>  
using std::\_\_is\_nothrow\_constructible\_impl = __is_nt_constructible_impl< __is_constructible(_Tp, _Args...),  
_Tp, _Args... >`
- `template<typename... >  
using std::\_\_void\_t = void`
- `template<typename... _Cond>  
using std::\_\_Require = typename enable_if< __and< _Cond... >::value >::type`
- `template<typename _Tp >  
using std::add\_const\_t = typename add_const< _Tp >::type`
- `template<typename _Tp >  
using std::add\_cv\_t = typename add_cv< _Tp >::type`
- `template<typename _Tp >  
using std::add\_lvalue\_reference\_t = typename add_lvalue_reference< _Tp >::type`
- `template<typename _Tp >  
using std::add\_pointer\_t = typename add_pointer< _Tp >::type`

- `template<typename _Tp >`  
using `std::add_rvalue_reference_t` = `typename add_rvalue_reference< _Tp >::type`
- `template<typename _Tp >`  
using `std::add_volatile_t` = `typename add_volatile< _Tp >::type`
- `template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>`  
using `std::aligned_storage_t` = `typename aligned_storage< _Len, _Align >::type`
- `template<size_t _Len, typename... _Types>`  
using `std::aligned_union_t` = `typename aligned_union< _Len, _Types... >::type`
- `template<typename... _Tp>`  
using `std::common_type_t` = `typename common_type< _Tp... >::type`
- `template<bool _Cond, typename _Iftrue, typename _Iffalse >`  
using `std::conditional_t` = `typename conditional< _Cond, _Iftrue, _Iffalse >::type`
- `template<typename _Tp >`  
using `std::decay_t` = `typename decay< _Tp >::type`
- `template<bool _Cond, typename _Tp = void>`  
using `std::enable_if_t` = `typename enable_if< _Cond, _Tp >::type`
- `typedef integral_constant< bool, false > std::false_type`
- `template<typename _Tp >`  
using `std::make_signed_t` = `typename make_signed< _Tp >::type`
- `template<typename _Tp >`  
using `std::make_unsigned_t` = `typename make_unsigned< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_all_extents_t` = `typename remove_all_extents< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_const_t` = `typename remove_const< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_cv_t` = `typename remove_cv< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_extent_t` = `typename remove_extent< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_pointer_t` = `typename remove_pointer< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_reference_t` = `typename remove_reference< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_volatile_t` = `typename remove_volatile< _Tp >::type`
- `template<typename _Tp >`  
using `std::result_of_t` = `typename result_of< _Tp >::type`
- `typedef integral_constant< bool, true > std::true_type`
- `template<typename _Tp >`  
using `std::underlying_type_t` = `typename underlying_type< _Tp >::type`
- `template<typename... >`  
using `std::void_t` = `void`

## Functions

- `template<typename _Fn, typename _Tp, typename... _Args>`  
constexpr bool `std::__call_is_nt` (`__invoke_memfun_ref`)
- `template<typename _Fn, typename _Tp, typename... _Args>`  
constexpr bool `std::__call_is_nt` (`__invoke_memfun_deref`)
- `template<typename _Fn, typename _Tp >`  
constexpr bool `std::__call_is_nt` (`__invoke_memobj_ref`)

- `template<typename _Fn, typename _Tp >`  
`constexpr bool std::__call_is_nt (__invoke_memobj_deref)`
- `template<typename _Fn, typename... _Args>`  
`constexpr bool std::__call_is_nt (__invoke_other)`
- `template<typename _Tp >`  
`auto std::declval () noexcept -> decltype(__declval< _Tp > (0))`
- `template<typename _Tp >`  
`enable_if< __and< __not< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_↵`  
`assignable< _Tp > >::value >::type std::swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_↵`  
`move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm>`  
`enable_if< __is_swappable< _Tp >::value >::type std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(_↵`  
`__is_nothrow_swappable< _Tp >::value)`

## Variables

- `template<typename _Tp >`  
`_GLIBCXX17_INLINE constexpr bool std::is_nothrow_swappable_v`
- `template<typename _Tp, typename _Up >`  
`_GLIBCXX17_INLINE constexpr bool std::is_nothrow_swappable_with_v`
- `template<typename _Tp >`  
`_GLIBCXX17_INLINE constexpr bool std::is_swappable_v`
- `template<typename _Tp, typename _Up >`  
`_GLIBCXX17_INLINE constexpr bool std::is_swappable_with_v`

## 6.648.1 Detailed Description

This is a Standard C++ Library header.

## 6.648.2 Macro Definition Documentation

### 6.648.2.1 \_GLIBCXX\_HAS\_NESTED\_TYPE

```
#define _GLIBCXX_HAS_NESTED_TYPE(  
    _NTYPE )
```

Use SFINAE to determine if the type `_Tp` has a publicly-accessible member type `_NTYPE`.

Definition at line 2354 of file `type_traits`.

## 6.649 type\_traits File Reference

### Classes

- struct `std::tr2::__reflection_typelist< _Elements >`
- struct `std::tr2::__reflection_typelist< _First, _Rest... >`
- struct `std::tr2::__reflection_typelist<>`
- struct `std::tr2::bases< _Tp >`
- struct `std::tr2::direct_bases< _Tp >`

## Namespaces

- [std](#)
- [std::tr2](#)

## Macros

- `#define _GLIBCXX_TR2_TYPE_TRAITS`

## 6.649.1 Detailed Description

This is a TR2 C++ Library header.

## 6.650 type\_traits File Reference

## Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_experimental_detect`
- `#define __cpp_lib_experimental_logical_traits`
- `#define __cpp_lib_experimental_type_trait_variable_templates`
- `#define _GLIBCXX_EXPERIMENTAL_TYPE_TRAITS`

## Typedefs

- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::detected_or = std::\_\_detected\_or< _Default, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::detected_or_t = typename std::detected\_or< _Default, _Op, _Args... >::type`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::detected_t = typename std::\_\_detector< nonesuch, void, _Op, ↔  
_Args... >::type`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::is_detected = typename std::\_\_detector< nonesuch, void, _Op,  
_Args... >::value_t`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::is_detected_convertible = is_convertible< detected_t< _Op, ↔  
_Args... >, _To >`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::is_detected_exact = is_same< _Expected, detected_t< _Op, ↔  
_Args... > >`
- `template<typename... >`  
`using std::experimental::fundamentals_v2::void_t = void`



## Variables

- `template<typename _Tp >`  
`constexpr size_t std::experimental::fundamentals_v1::alignment_of_v`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::fundamentals_v2::conjunction_v`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::fundamentals_v2::disjunction_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t std::experimental::fundamentals_v1::extent_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::has_virtual_destructor_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_abstract_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_arithmetic_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_array_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_assignable_v`
- `template<typename _Base, typename _Derived >`  
`constexpr bool std::experimental::fundamentals_v1::is_base_of_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_class_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_compound_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_const_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v1::is_constructible_v`
- `template<typename _From, typename _To >`  
`constexpr bool std::experimental::fundamentals_v1::is_convertible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_copy_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_copy_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_default_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_destructible_v`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v2::is_detected_convertible_v`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v2::is_detected_exact_v`
- `template<template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v2::is_detected_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_empty_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_enum_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_final_v`

- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_floating_point_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_function_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_fundamental_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_integral_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_literal_type_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_lvalue_reference_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_member_function_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_member_object_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_member_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_move_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_move_constructible_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_copy_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_copy_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_default_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_destructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_move_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_move_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_null_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_object_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_pod_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_polymorphic_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_reference_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_rvalue_reference_v`

- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_same_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_scalar_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_signed_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_standard_layout_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivial_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_copy_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_copy_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_copyable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_default_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_destructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_move_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_move_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_union_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_unsigned_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_void_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_volatile_v`
- `template<typename _Pp >`  
`constexpr bool std::experimental::fundamentals_v2::negation_v`
- `template<typename _Tp >`  
`constexpr size_t std::experimental::fundamentals_v1::rank_v`

#### 6.650.1 Detailed Description

This is a TS C++ Library header.

## 6.651 `type_traits.h` File Reference

### Namespaces

- [`\_\_gnu\_cxx`](#)

## Functions

- `template<typename _Type >`  
`bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type >`  
`bool __gnu_cxx::__is_null_pointer (_Type)`
- `bool __gnu_cxx::__is_null_pointer (std::nullptr_t)`

### 6.651.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.652 type\_utils.hpp File Reference

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

### Typedefs

- `typedef std::tr1::integral_constant< int, 0 > __gnu_pbds::detail::false_type`
- `typedef std::tr1::integral_constant< int, 1 > __gnu_pbds::detail::true_type`

### 6.652.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

## 6.653 typeindex File Reference

### Classes

- struct [std::hash< \\_Tp >](#)
- struct [std::hash< type\\_index >](#)
- struct [std::type\\_index](#)

### Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_TYPEINDEX`

### 6.653.1 Detailed Description

This is a Standard C++ Library header.

## 6.654 typeinfo File Reference

### Classes

- class [std::bad\\_cast](#)
- class [std::bad\\_typeid](#)
- class [std::type\\_info](#)

### Namespaces

- [std](#)

## Macros

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`
- `#define _TYPEINFO`

### 6.654.1 Detailed Description

This is a Standard C++ Library header.

## 6.655 typelist.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::typelist](#)

## Macros

- `#define _GLIBCXX_TPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TPELIST_CHAIN16(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)`
- `#define _GLIBCXX_TPELIST_CHAIN17(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)`
- `#define _GLIBCXX_TPELIST_CHAIN18(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)`
- `#define _GLIBCXX_TPELIST_CHAIN19(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)`
- `#define _GLIBCXX_TPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TPELIST_CHAIN20(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)`
- `#define _GLIBCXX_TPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

## Functions

- `template<typename Fn, typename Typelist >`  
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Gn, typename Typelist >`  
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV >`  
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist >`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV >`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`

## 6.655.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

## 6.656 types.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef int64\_t [\\_\\_gnu\\_parallel::\\_CASable](#)
- typedef uint64\_t [\\_\\_gnu\\_parallel::\\_SequenceIndex](#)
- typedef uint16\_t [\\_\\_gnu\\_parallel::\\_ThreadIndex](#)

### Enumerations

- enum [\\_\\_gnu\\_parallel::\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_\\_gnu\\_parallel::\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_\\_gnu\\_parallel::\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_\\_gnu\\_parallel::\\_Parallelism](#) { [\\_\\_gnu\\_parallel::sequential](#), [\\_\\_gnu\\_parallel::parallel\\_unbalanced](#), [\\_\\_gnu\\_parallel::parallel\\_balanced](#), [\\_\\_gnu\\_parallel::parallel\\_omp](#), [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop\\_static](#), [\\_\\_gnu\\_parallel::parallel\\_taskqueue](#) }
- enum [\\_\\_gnu\\_parallel::\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_\\_gnu\\_parallel::\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_\\_gnu\\_parallel::\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

### Variables

- static const int [\\_\\_gnu\\_parallel::\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_CASable\\_mask](#)

#### 6.656.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

## 6.657 types\_traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, Mapped >](#)
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_hash< \\_Th >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_value< \\_Tv >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, Mapped, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, null\\_type, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_base< Key, null\\_type, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::type\\_dispatch< Key, Mapped, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::types\\_traits< Key, Mapped, \\_Alloc, Store\\_Hash >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 6.657.1 Detailed Description

Contains a traits class of types used by containers.

## 6.658 uniform\_int\_dist.h File Reference

## Classes

- class [std::uniform\\_int\\_distribution<\\_IntType>](#)
- struct [std::uniform\\_int\\_distribution<\\_IntType>::param\\_type](#)

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<typename _Tp>  
bool std::\_\_detail::\_Power\_of\_2 (_Tp __x)`

## 6.658.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 6.659 unique\_copy.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result, \_↵  
_BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result)`



### 6.659.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

## 6.660 `unique_ptr.h` File Reference

### Classes

- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- struct `std::hash<unique_ptr<_Tp, _Dp>>`
- class `std::unique_ptr<_Tp, _Dp>`
- class `std::unique_ptr<_Tp[], _Dp>`

### Namespaces

- `std`

### Macros

- `#define __cpp_lib_make_unique`

### Functions

- `template<typename _Tp, typename... _Args>  
_MakeUniq<_Tp>::__single_object std::make_unique (_Args &&... __args)`
- `template<typename _Tp>  
_MakeUniq<_Tp>::__array std::make_unique (size_t __num)`
- `template<typename _Tp, typename... _Args>  
_MakeUniq<_Tp>::__invalid_type std::make_unique (_Args &&...)=delete`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >  
bool std::operator!= (const unique_ptr<_Tp, _Dp> &__x, const unique_ptr<_Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp >  
bool std::operator!= (const unique_ptr<_Tp, _Dp> &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >  
bool std::operator!= (nullptr_t, const unique_ptr<_Tp, _Dp> &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >  
bool std::operator< (const unique_ptr<_Tp, _Dp> &__x, const unique_ptr<_Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp >  
bool std::operator< (const unique_ptr<_Tp, _Dp> &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >  
bool std::operator< (nullptr_t, const unique_ptr<_Tp, _Dp> &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >  
bool std::operator<= (const unique_ptr<_Tp, _Dp> &__x, const unique_ptr<_Up, _Ep> &__y)`
- `template<typename _Tp, typename _Dp >  
bool std::operator<= (const unique_ptr<_Tp, _Dp> &__x, nullptr_t)`

- `template<typename _Tp, typename _Dp >`  
`bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp >`  
`enable_if< __is_swappable< _Dp >::value >::type std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp, typename _Dp >`  
`enable_if< !__is_swappable< _Dp >::value >::type std::swap (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete`

### 6.660.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 6.661 unordered\_base.h File Reference

### Namespaces

- [std](#)
- [std::\\_\\_profile](#)

### Functions

- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>`  
`bool std::__profile::__are_equal (const _UnorderedCont &__uc, const __detail::__Hash_node< _Value, __Cache_hash_code > * __lhs, const __detail::__Hash_node< _Value, __Cache_hash_code > * __rhs)`
- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>`  
`std::size_t std::__profile::__get_bucket_index (const _UnorderedCont &__uc, const __detail::__Hash_node< _Value, __Cache_hash_code > * __node)`

### 6.661.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.662 unordered\_map File Reference

### Macros

- `#define _GLIBCXX_UNORDERED_MAP`

### 6.662.1 Detailed Description

This is a Standard C++ Library header.

## 6.663 unordered\_map File Reference

### Classes

- class `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`

### Namespaces

- `std`
- `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_MAP`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
bool std::__debug::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
bool std::__debug::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
bool std::__debug::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
bool std::__debug::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
void std::__debug::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key,  
_Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
void std::__debug::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_  
multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

## 6.663.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.664 unordered\_map File Reference

## Classes

- class [std::\\_\\_profile::unordered\\_map<\\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)
- class [std::\\_\\_profile::unordered\\_multimap<\\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Macros

- `#define GLIBCXX_BASE`
- `#define GLIBCXX_BASE`
- `#define GLIBCXX_PROFILE_UNORDERED_MAP`
- `#define GLIBCXX_STD_BASE`
- `#define GLIBCXX_STD_BASE`

## Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__profile::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__profile::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__profile::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__profile::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &__x, const  
unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
void std::__profile::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map<_Key,  
_Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
void std::__profile::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_↵  
multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

#### 6.664.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

### 6.665 unordered\_map File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_MAP`

#### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>>`  
`using std::experimental::fundamentals_v2::pmr::unordered_map = std::unordered\_map<_Key, _Tp, _Hash,`  
`_Pred, polymorphic_allocator< pair< const _Key, _Tp > >>`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>>`  
`using std::experimental::fundamentals_v2::pmr::unordered_multimap = std::unordered\_multimap<_Key, ↵`  
`_Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > >>`

#### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc >`  
`&__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc`  
`> &__cont, _Predicate __pred)`

#### 6.665.1 Detailed Description

This is a TS C++ Library header.

### 6.666 unordered\_map.h File Reference

#### Classes

- class [std::unordered\\_map](#)<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multimap](#)<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multimap](#)<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>  
using std::__umap_hashtable = _Hashtable<_Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::__umap_traits = __detail::_Hashtable_traits<_Cache, false, true >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>  
using std::__ummap_hashtable = _Hashtable<_Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::__ummap_traits = __detail::_Hashtable_traits<_Cache, false, false >`

## Functions

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >  
bool std::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >  
bool std::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >  
bool std::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >  
bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >  
void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >  
void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

## 6.666.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

## 6.667 unordered\_set File Reference

### Macros

- `#define _GLIBCXX_UNORDERED_SET`

### 6.667.1 Detailed Description

This is a Standard C++ Library header.

## 6.668 unordered\_set File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)
- class [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)

### Namespaces

- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_SET`

### Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
void std::__debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
void std::__debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

## 6.668.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.669 unordered\_set File Reference

## Classes

- class [std::\\_\\_profile::unordered\\_multiset<\\_Value, \\_Hash, \\_Pred, \\_Alloc>](#)
- class [std::\\_\\_profile::unordered\\_set<\\_Key, \\_Hash, \\_Pred, \\_Alloc>](#)

## Namespaces

- [std](#)
- [std::\\_\\_profile](#)

## Macros

- `#define GLIBCXX_BASE`
- `#define GLIBCXX_BASE`
- `#define GLIBCXX_PROFILE_UNORDERED_SET`
- `#define GLIBCXX_STD_BASE`
- `#define GLIBCXX_STD_BASE`

## Functions

- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__profile::operator!= (const unordered_set<_Key, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Key, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__profile::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__profile::operator== (const unordered_set<_Key, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Key, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__profile::operator== (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__profile::swap (unordered_set<_Key, _Hash, _Pred, _Alloc> &__x, unordered_set<_Key, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__profile::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`



#### 6.669.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

### 6.670 unordered\_set File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_SET`

#### Typedefs

- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **`std::experimental::fundamentals_v2::pmr::unordered_multiset`** = [std::unordered\\_multiset](#)< \_Key, \_Hash, \_Pred, polymorphic\_allocator< \_Key > >
- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **`std::experimental::fundamentals_v2::pmr::unordered_set`** = [std::unordered\\_set](#)< \_Key, \_Hash, \_Pred, polymorphic\_allocator< \_Key > >

#### Functions

- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (unordered\_set< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (unordered\_multiset< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

#### 6.670.1 Detailed Description

This is a TS C++ Library header.

### 6.671 unordered\_set.h File Reference

#### Classes

- class [std::unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_set](#)< \_Value, \_Hash, \_Pred, \_Alloc >

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>  
using std::__umset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::__umset_traits = __detail::_Hashtable_traits<_Cache, true, false >`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>  
using std::__uset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::__uset_traits = __detail::_Hashtable_traits<_Cache, true, true >`

## Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator!= (const unordered_set<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator== (const unordered_set<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator== (const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
void std::swap (unordered_set<_Value, _Hash, _Pred, _Alloc > &__x, unordered_set<_Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
void std::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

## 6.671.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

## 6.672 update\_fn\_imps.hpp File Reference

## 6.672.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 6.673 utility File Reference

### Classes

- struct [std::\\_\\_is\\_tuple\\_like\\_impl](#)< std::pair< \_T1, \_T2 > >
- struct [std::integer\\_sequence](#)< \_Tp, \_Idx >
- struct [std::tuple\\_element](#)< \_Int, \_Tp >
- struct [std::tuple\\_element](#)< 0, std::pair< \_Tp1, \_Tp2 > >
- struct [std::tuple\\_element](#)< 1, std::pair< \_Tp1, \_Tp2 > >
- struct [std::tuple\\_size](#)< \_Tp >
- struct [std::tuple\\_size](#)< std::pair< \_Tp1, \_Tp2 > >

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_exchange_function`
- `#define __cpp_lib_integer_sequence`
- `#define __cpp_lib_tuple_element_t`
- `#define __cpp_lib_tuples_by_type`
- `#define _GLIBCXX_UTILITY`

### Typedefs

- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up><↵  
::value>::type, size_t = tuple_size<_Tp>::value>  
using std::\_\_enable\_if\_has\_tuple\_size = _Tp`
- `template<std::size_t __i, typename _Tp >  
using std::\_\_tuple\_element\_t = typename tuple_element< __i, _Tp >::type`
- `template<size_t... _Idx>  
using std::index\_sequence = integer_sequence< size_t, _Idx... >`
- `template<typename... _Types>  
using std::index\_sequence\_for = make_index_sequence< sizeof...(_Types)>`
- `template<size_t _Num>  
using std::make\_index\_sequence = make_integer_sequence< size_t, _Num >`
- `template<typename _Tp, _Tp _Num>  
using std::make\_integer\_sequence = integer_sequence< _Tp, __integer_pack(_Num)... >`
- `template<std::size_t __i, typename _Tp >  
using std::tuple\_element\_t = typename tuple_element< __i, _Tp >::type`

## Functions

- `template<typename _Tp, typename _Up = _Tp>  
_Tp std::exchange (_Tp &__obj, _Up &&__new_val)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (std::pair< _Tp1, _Tp2 > &__in)  
noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type && std::get (std::pair< _Tp1, _Tp2 > &&__in)  
noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (const std::pair< _Tp1, _Tp2  
> &__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type && std::get (const std::pair< _Tp1, _Tp2  
> &&__in) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp & std::get (pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp & std::get (const pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp && std::get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp && std::get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp & std::get (pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp & std::get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp && std::get (pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp && std::get (const pair< _Up, _Tp > &&__p) noexcept`

## 6.673.1 Detailed Description

This is a Standard C++ Library header.

## 6.674 utility File Reference

## Namespaces

- [std](#)

## Macros

- `#define \_GLIBCXX\_EXPERIMENTAL\_UTILITY`

## Typedefs

- using **std::experimental::fundamentals\_v2::erased\_type** = std::\_\_erased\_type

## 6.674.1 Detailed Description

This is a TS C++ Library header.

## 6.675 valarray File Reference

### Classes

- class [std::gslice\\_array<\\_Tp>](#)
- class [std::indirect\\_array<\\_Tp>](#)
- class [std::mask\\_array<\\_Tp>](#)
- class [std::slice\\_array<\\_Tp>](#)
- class [std::valarray<\\_Tp>](#)
- class [std::valarray<\\_Tp>](#)

### Namespaces

- [std](#)

### Macros

- **#define \_DEFINE\_BINARY\_OPERATOR**(\_Op, \_Name)
- **#define \_DEFINE\_VALARRAY\_AUGMENTED\_ASSIGNMENT**(\_Op, \_Name)
- **#define \_DEFINE\_VALARRAY\_EXPR\_AUGMENTED\_ASSIGNMENT**(\_Op, \_Name)
- **#define \_DEFINE\_VALARRAY\_UNARY\_OPERATOR**(\_Op, \_Name)
- **#define \_GLIBCXX\_VALARRAY**

### Functions

- template<class \_Tp>  
\_Tp \* [std::begin](#) (valarray<\_Tp> &\_\_va)
- template<class \_Tp>  
const \_Tp \* [std::begin](#) (const valarray<\_Tp> &\_\_va)
- template<class \_Tp>  
\_Tp \* [std::end](#) (valarray<\_Tp> &\_\_va)
- template<class \_Tp>  
const \_Tp \* [std::end](#) (const valarray<\_Tp> &\_\_va)
- template<typename \_Tp>  
\_Expr<\_BinClos<\_\_not\_equal\_to, \_ValArray, \_Constant, \_Tp, \_Tp>, typename \_\_fun<\_\_not\_equal\_to, \_Tp>::result\_type> **std::operator!=** (const valarray<\_Tp> &\_\_v, const \_Tp &\_\_t)

- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp`  
`>::result_type > std::operator!= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp`  
`>::result_type > std::operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __modulus, _Tp >::result_`  
`<_type > std::operator% (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __modulus, _Tp >::`  
`result_type > std::operator% (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __modulus, _Tp >::`  
`result_type > std::operator% (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp`  
`>::result_type > std::operator& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp`  
`>::result_type > std::operator& (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp`  
`>::result_type > std::operator& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > std::operator&& (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > std::operator&& (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > std::operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::`  
`result_type > std::operator* (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::`  
`result_type > std::operator* (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::`  
`result_type > std::operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type >`  
`std::operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type >`  
`std::operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type >`  
`std::operator+ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type`  
`> std::operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type`  
`> std::operator- (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type`  
`> std::operator- (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type`  
`> std::operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_↵`  
`type > std::operator/ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_↵`  
`type > std::operator/ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`std::operator< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`std::operator< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`std::operator< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_↵`  
`type > std::operator<< (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_↵`  
`type > std::operator<< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_↵`  
`type > std::operator<< (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >↵`  
`::result_type > std::operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less_equal, _Tp >↵`  
`::result_type > std::operator<= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >↵`  
`::result_type > std::operator<= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_↵`  
`type > std::operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >↵`  
`::result_type > std::operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >↵`  
`::result_type > std::operator== (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_↵`  
`type > std::operator> (const valarray< _Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type`  
`> std::operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type`  
`< type > std::operator> (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp`  
`>::result_type > std::operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp`  
`>::result_type > std::operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp`  
`>::result_type > std::operator>= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >`  
`::result_type > std::operator>> (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_right, _Tp >`  
`::result_type > std::operator>> (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >`  
`::result_type > std::operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >`  
`::result_type > std::operator^ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >`  
`::result_type > std::operator^ (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >`  
`::result_type > std::operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >`  
`::result_type > std::operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >`  
`::result_type > std::operator| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >`  
`::result_type > std::operator| (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >`  
`::result_type > std::operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_or, _Tp >`  
`::result_type > std::operator|| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >`  
`::result_type > std::operator|| (const _Tp &__t, const valarray< _Tp > &__v)`



### 6.675.1 Detailed Description

This is a Standard C++ Library header.

### 6.676 valarray\_after.h File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define _DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define _DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

#### Functions

- `template<class _Dom >  
_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > std::abs (const _Expr< _Dom, type-  
name _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `template<class _Dom >  
_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type > std::acos (const _Expr< _Dom,  
typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom >  
_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > std::asin (const _Expr< _Dom, type-  
name _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom >  
_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr< _Dom,  
typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::atan2 (const  
_Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type  
> &__e2)`
- `template<class _Dom >  
_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_←  
_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _←  
Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`  
`> std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`> std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`  
`> std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v,`  
`const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type > std::cosh (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > std::exp (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > std::log (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type > std::log10 (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const valarray< typename _`  
`Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __not_equal_to, type-`  
`name _Dom1::value_type >::result_type > std::operator!= (const _Expr< _Dom1, typename _Dom1::value_`  
`type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __modulus, typename _`  
`Dom1::value_type >::result_type > std::operator% (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__modulus, typename _Dom::value_type >::result_type > std::operator% (const typename _Dom::value_type`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__modulus, typename _Dom::value_type >::result_type > std::operator% (const valarray< typename _Dom`  
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const typename _Dom::value_`  
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __`  
`fun< __bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_and, type-`  
`name _Dom1::value_type >::result_type > std::operator& (const _Expr< _Dom1, typename _Dom1::value_`  
`type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_and, typename`  
`_Dom1::value_type >::result_type > std::operator&& (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const typename _Dom::value_↵`  
`_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const valarray< typename _↵`  
`Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __multiplies, typename`  
`_Dom1::value_type >::result_type > std::operator* (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr< _Dom, typename _↵`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > std::operator* (const typename _Dom::value_↵`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr< _Dom, typename _↵`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__multiplies, typename _Dom::value_type >::result_type > std::operator* (const valarray< typename _Dom_↵`  
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom, typename _Dom::value_↵`  
`type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom, typename _Dom::value_↵`  
`type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > std::operator+ (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1<`  
`::value_type >::result_type > std::operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus,`  
`typename _Dom::value_type >::result_type > std::operator+ (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __`  
`minus, typename _Dom::value_type >::result_type > std::operator- (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __`  
`minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom, typename _Dom<`  
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __`  
`minus, typename _Dom::value_type >::result_type > std::operator- (const valarray< typename _Dom::value<`  
`_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1<`  
`::value_type >::result_type > std::operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__`  
`v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __`  
`minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom, typename _Dom<`  
`::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename <`  
`Dom1::value_type >::result_type > std::operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__`  
`v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __`  
`divides, typename _Dom::value_type >::result_type > std::operator/ (const _Expr< _Dom, typename _Dom<`  
`::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __`  
`divides, typename _Dom::value_type >::result_type > std::operator/ (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __`  
`divides, typename _Dom::value_type >::result_type > std::operator/ (const _Expr< _Dom, typename _Dom<`  
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __`  
`divides, typename _Dom::value_type >::result_type > std::operator/ (const valarray< typename _Dom::value<`  
`_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1<`  
`::value_type >::result_type > std::operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > std::operator< (const _Expr< _Dom, typename _Dom::value_↵`  
`_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > std::operator< (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > std::operator< (const _Expr< _Dom, typename _Dom::value_↵`  
`_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less,`  
`typename _Dom::value_type >::result_type > std::operator< (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_left, typename _↵`  
`_Dom1::value_type >::result_type > std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< ↵`  
`__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< ↵`  
`__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const typename _Dom::value_type`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< ↵`  
`__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< ↵`  
`__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const valarray< typename _Dom_↵`  
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less_equal, typename`  
`_Dom1::value_type >::result_type > std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const typename _Dom::value_↵`  
`_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const valarray< typename _`  
`Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`   
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const valarray< typename _Dom<`  
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`   
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const typename _Dom::value_type`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __equal_to, typename`   
`_Dom1::value_type >::result_type > std::operator== (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`   
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`   
`__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom, typename`   
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`   
`__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom, typename _Dom<`  
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`   
`__greater, typename _Dom::value_type >::result_type > std::operator> (const valarray< typename _Dom<`  
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater, typename`   
`_Dom1::value_type >::result_type > std::operator> (const _Expr< _Dom1, typename _Dom1::value_type >`  
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`   
`__greater, typename _Dom::value_type >::result_type > std::operator> (const typename _Dom::value_type`  
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __`  
`fun< __greater_equal, typename _Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`   
`__fun< __greater_equal, typename _Dom::value_type >::result_type > std::operator>= (const typename`   
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__greater_equal, typename _Dom::value_type >::result_type > std::operator>=` (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, type-`  
`name _Dom1::value_type >::result_type > std::operator>=` (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__greater_equal, typename _Dom::value_type >::result_type > std::operator>=` (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_right, typename`  
`_Dom1::value_type >::result_type > std::operator>>` (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename`  
`_Dom1::value_type >::result_type > std::operator^` (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)



- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename`  
`_Dom1::value_type >::result_type > std::operator| (const _Expr< _Dom1, typename _Dom1::value_type >`  
`& __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr< _Dom, typename`  
`_Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr< _Dom, typename`  
`_Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const typename _Dom::value_type`  
`& __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const valarray< typename _Dom`  
`::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const valarray< typename _Dom`  
`::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`  
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr< _Dom, typename`  
`_Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`  
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr< _Dom, typename`  
`_Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`  
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const typename _Dom::value_type`  
`& __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename`  
`_Dom1::value_type >::result_type > std::operator|| (const _Expr< _Dom1, typename _Dom1::value_type >`  
`& __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type`  
`> std::pow (const valarray< typename _Dom::valarray > & __v, const _Expr< _Dom, typename _Dom::value`  
`_type > & __e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value`  
`_type > std::pow (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type >`  
`& __e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp > & __v,`  
`const _Tp & __t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value`  
`_type > std::pow (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type >`  
`& __e)`

- ```
type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type
&__t)
```
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp > &__v,`  
`const valarray< _Tp > &__w)`
  - `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type`  
`> std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::`  
`::value_type > &__v)`
  - `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::pow (const`  
`_Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type`  
`> &__e2)`
  - `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow (const _Tp &__t, const valarray<`  
`_Tp > &__v)`
  - `template<class _Dom >`  
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > std::sin (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
  - `template<typename _Tp >`  
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > &__v)`
  - `template<class _Dom >`  
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > std::sinh (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
  - `template<typename _Tp >`  
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > &__v)`
  - `template<typename _Tp >`  
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > &__v)`
  - `template<class _Dom >`  
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > std::sqrt (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
  - `template<typename _Tp >`  
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray< _Tp > &__v)`
  - `template<class _Dom >`  
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > std::tan (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
  - `template<class _Dom >`  
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type > std::tanh (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
  - `template<typename _Tp >`  
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray< _Tp > &__v)`

### 6.676.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.677 valarray\_array.h File Reference

### Namespaces

- [std](#)

## Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

## Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp >`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_and ( _Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_and ( _Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_bitwise_and ( _Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_and ( _Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_or ( _Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_xor ( _Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_xor ( _Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_bitwise_xor ( _Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_xor ( _Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_bitwise_xor ( _Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`



- `template<typename _Tp >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_plus (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`

### 6.677.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.678 valarray\_array.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define VALARRAY_ARRAY_TCC`

## Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`

### 6.678.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.679 valarray\_before.h File Reference

### Namespaces

- [std](#)

### 6.679.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 6.680 vector File Reference

## Macros

- `#define _GLIBCXX_VECTOR`

## 6.680.1 Detailed Description

This is a Standard C++ Library header.

## 6.681 vector File Reference

## Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_vector<\\_SafeSequence, \\_BaseSequence >](#)
- class [std::\\_\\_debug::vector<\\_Tp, \\_Allocator >](#)
- struct [std::hash<\\_\\_debug::vector<bool, \\_Alloc > >](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_VECTOR`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(*conditional */)`

### 6.681.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 6.682 vector File Reference

### Classes

- struct [std::hash< \\_\\_profile::vector< bool, \\_Alloc > >](#)

### Namespaces

- [std](#)
- [std::\\_\\_profile](#)

### Macros

- `#define _GLIBCXX_PROFILE_VECTOR`

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(*conditional  
*)`

### 6.682.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

## 6.683 vector File Reference

### Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_experimental_erase_if`
- `#define _GLIBCXX_EXPERIMENTAL_VECTOR`

## Typedefs

- `template<typename _Tp >`  
using `std::experimental::fundamentals_v2::pmr::vector` = `std::vector`< `_Tp`, `polymorphic_allocator`< `_Tp` >  
>

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >`  
void `std::experimental::fundamentals_v2::erase` (`vector`< `_Tp`, `_Alloc` > &\_\_cont, const `_Up` &\_\_value)
- `template<typename _Tp, typename _Alloc, typename _Predicate >`  
void `std::experimental::fundamentals_v2::erase_if` (`vector`< `_Tp`, `_Alloc` > &\_\_cont, `_Predicate` \_\_pred)

### 6.683.1 Detailed Description

This is a TS C++ Library header.

## 6.684 vector.tcc File Reference

### Namespaces

- `std`

## Macros

- `#define _VECTOR_TCC`

### 6.684.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 6.685 vstring.h File Reference

### Classes

- class `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` >
- struct `std::hash`< `__gnu_cxx::__u16vstring` >
- struct `std::hash`< `__gnu_cxx::__u32vstring` >
- struct `std::hash`< `__gnu_cxx::__vstring` >
- struct `std::hash`< `__gnu_cxx::__wvstring` >



- [illegible]



- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT  
*__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >  
&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string<  
\_CharT, \_Traits, \_Alloc, \_Base > &\_\_str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
void \_\_gnu\_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`

### 6.685.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 6.686 vstring.tcc File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define VSTRING\_TCC`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string<  
\_CharT, \_Traits, \_Alloc, \_Base > &\_\_str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const __versa_string< _CharT, _Traits,  
_Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const _CharT *__lhs, const __versa_  
__string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (_CharT __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const __versa_string< _CharT, _Traits,  
_Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const __versa_string< _CharT, _Traits,  
_Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string<  
\_CharT, \_Traits, \_Alloc, \_Base > &\_\_str)`

## 6.686.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

6.687 `vstring_fwd.h` File Reference

## Classes

- class `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>`
- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`

## Namespaces

- `__gnu_cxx`

## Typedefs

- typedef `__versa_string< char, std::char\_traits< char >, std::allocator< char >, __rc_string_base > __gnu_cxx::__rc_string`
- typedef `__vstring __gnu_cxx::__sso_string`
- typedef `__versa_string< char16_t, std::char\_traits< char16\_t >, std::allocator< char16\_t >, __rc_string_base > __gnu_cxx::__u16rc_string`
- typedef `__u16vstring __gnu_cxx::__u16sso_string`
- typedef `__versa_string< char16_t > __gnu_cxx::__u16vstring`
- typedef `__versa_string< char32_t, std::char\_traits< char32\_t >, std::allocator< char32\_t >, __rc_string_base > __gnu_cxx::__u32rc_string`
- typedef `__u32vstring __gnu_cxx::__u32sso_string`
- typedef `__versa_string< char32_t > __gnu_cxx::__u32vstring`
- typedef `__versa_string< char > __gnu_cxx::__vstring`
- typedef `__versa_string< wchar_t, std::char\_traits< wchar\_t >, std::allocator< wchar\_t >, __rc_string_base > __gnu_cxx::__wrc_string`
- typedef `__wvstring __gnu_cxx::__wsso_string`
- typedef `__versa_string< wchar_t > __gnu_cxx::__wvstring`

## 6.687.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

6.688 `vstring_util.h` File Reference

## Namespaces

- `__gnu_cxx`

### 6.688.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 6.689 workstealing.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_Job<\\_DifferenceTp >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- `#define _GLIBCXX_JOB_VOLATILE`

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_workstealing (_RAIter __begin, _RAIter __end, ↵  
_Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter >↵  
::difference_type __bound)`

### 6.689.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.

Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

## Index

- `_AlgorithmStrategy`
  - `__gnu_parallel`, [503](#)
- `_BALLOC_ALIGN_BYTES`
  - `bitmap_allocator.h`, [4020](#)
- `_BinIndex`
  - `__gnu_parallel`, [503](#)
- `_Bit_scan_forward`
  - `__gnu_cxx`, [463](#)
- `_CASable`
  - `__gnu_parallel`, [503](#)
- `_CASable_bits`
  - `__gnu_parallel`, [554](#)
- `_CASable_mask`
  - `__gnu_parallel`, [555](#)
- `_Construct`
  - `std`, [686](#)
- `_DRandomShufflingGlobalData`
  - `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter >`, [1245](#)
- `_Destroy`
  - `std`, [686](#), [687](#)
- `_Destroy_n`
  - `std`, [687](#)
- `_Distance_precision`
  - `__gnu_debug`, [489](#)
- `_FindAlgorithm`
  - `__gnu_parallel`, [504](#)
- `_Find_first`
  - `SGI`, [375](#)
- `_Find_next`
  - `SGI`, [376](#)
- `_GLIBCXX_BAL_QUICKSORT`
  - `features.h`, [4109](#)
- `_GLIBCXX_CALL`
  - `completime_settings.h`, [4052](#)
- `_GLIBCXX_DEBUG_VERIFY_AT`
  - `macros.h`, [4188](#)
- `_GLIBCXX_DEQUE_BUF_SIZE`
  - `stl_deque.h`, [4355](#)
- `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
  - `features.h`, [4110](#)
- `_GLIBCXX_FIND_EQUAL_SPLIT`
  - `features.h`, [4110](#)
- `_GLIBCXX_FIND_GROWING_BLOCKS`
  - `features.h`, [4110](#)
- `_GLIBCXX_HAS_NESTED_TYPE`
  - `type_traits`, [4408](#)
- `_GLIBCXX_MERGESORT`
  - `features.h`, [4110](#)
- `_GLIBCXX_PARALLEL_ASSERTIONS`
  - `completime_settings.h`, [4052](#)
- `_GLIBCXX_PARALLEL_CONDITION`
  - `settings.h`, [4315](#)
- `_GLIBCXX_PARALLEL_LENGTH`
  - `multiway_merge.h`, [4207](#)
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA`
  - `__gnu_profile`, [561](#)
- `_GLIBCXX_QUICKSORT`
  - `features.h`, [4111](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
  - `completime_settings.h`, [4052](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
  - `completime_settings.h`, [4053](#)
- `_GLIBCXX_SCALE_DOWN_FPU`
  - `completime_settings.h`, [4053](#)
- `_GLIBCXX_TREE_DYNAMIC_BALANCING`
  - `features.h`, [4111](#)
- `_GLIBCXX_TREE_FULL_COPY`
  - `features.h`, [4111](#)
- `_GLIBCXX_TREE_INITIAL_SPLITTING`
  - `features.h`, [4111](#)
- `_GLIBCXX_VERBOSE_LEVEL`
  - `completime_settings.h`, [4053](#)
- `_GLIBCXX_VOLATILE`
  - `partition.h`, [4236](#)
  - `queue.h`, [4261](#)
- `_GuardedIterator`
  - `__gnu_parallel::_GuardedIterator<_RAIter, _Compare >`, [1253](#)
- `_LoserTreeBase`
  - `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >`, [1272](#)
- `_M_allocate_and_copy`
  - `std::vector<_Tp, _Alloc >`, [3911](#)
- `_M_allocate_single_object`
  - `__gnu_cxx::bitmap_allocator<_Tp >`, [973](#)
- `_M_attach`
  - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >`, [1119](#)
  - `__gnu_debug::_Safe_iterator_base`, [1130](#)
  - `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence >`, [1137](#)
  - `__gnu_debug::_Safe_local_iterator_base`, [1147](#)
- `_M_attach_single`
  - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >`, [1119](#), [1120](#)
  - `__gnu_debug::_Safe_iterator_base`, [1130](#)
  - `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence >`, [1137](#), [1138](#)
  - `__gnu_debug::_Safe_local_iterator_base`, [1147](#)
- `_M_attached_to`

- `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1120
- `__gnu_debug::Safe_iterator_base`, 1130
- `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1138
- `__gnu_debug::Safe_local_iterator_base`, 1147
- `_M_before_dereferenceable`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1120
- `_M_begin`
  - `__gnu_parallel::Piece< _DifferenceTp >`, 1290
- `_M_bin_proc`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 1245
- `_M_bins_begin`
  - `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 1247
- `_M_buf`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1003
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1067
  - `std::basic_filebuf< _CharT, _Traits >`, 1988
- `_M_buf_locale`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1003
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1067
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1091
  - `std::basic_filebuf< _CharT, _Traits >`, 1988
  - `std::basic_streambuf< _CharT, _Traits >`, 2484
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2573
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3954
- `_M_buf_size`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1003
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1067
  - `std::basic_filebuf< _CharT, _Traits >`, 1988
- `_M_can_compare`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1120
  - `__gnu_debug::Safe_iterator_base`, 1130
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1138
  - `__gnu_debug::Safe_local_iterator_base`, 1147
- `_M_clear`
  - `__gnu_cxx::free_list`, 1011
- `_M_comp`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, 1274
- `_M_const_iterators`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1114
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1153
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1157
  - `__gnu_debug::Safe_sequence_base`, 1161
- `__gnu_debug::Safe_unordered_container< _Container >`, 1165
- `__gnu_debug::Safe_unordered_container_base`, 1169
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1200
- `std::__debug::deque< _Tp, _Allocator >`, 1703
- `std::__debug::forward_list< _Tp, _Alloc >`, 1708
- `std::__debug::list< _Tp, _Allocator >`, 1713
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 1718
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 1724
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 1729
- `std::__debug::set< _Key, _Compare, _Allocator >`, 1734
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1740
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1746
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 1752
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 1758
- `std::__debug::vector< _Tp, _Allocator >`, 1764
- `_M_const_local_iterators`
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1165
  - `__gnu_debug::Safe_unordered_container_base`, 1169
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1740
  - `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1746
  - `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 1752
  - `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 1758
- `_M_create_node`
  - `std::list< _Tp, _Alloc >`, 3164
- `_M_create_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 983
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1046
  - `std::basic_filebuf< _CharT, _Traits >`, 1968
- `_M_deallocate_single_object`
  - `__gnu_cxx::bitmap_allocator< _Tp >`, 974
- `_M_dereferenceable`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1121
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1138
- `_M_destroy_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 984

- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1047
- `std::basic_filebuf< _CharT, _Traits >`, 1968
- `_M_detach`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1121
  - `__gnu_debug::Safe_iterator_base`, 1130
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1139
  - `__gnu_debug::Safe_local_iterator_base`, 1147
- `_M_detach_all`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1112
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1151
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1155
  - `__gnu_debug::Safe_sequence_base`, 1159
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1163
  - `__gnu_debug::Safe_unordered_container_base`, 1167
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1176
  - `std::__debug::deque< _Tp, _Allocator >`, 1701
  - `std::__debug::forward_list< _Tp, _Alloc >`, 1706
  - `std::__debug::list< _Tp, _Allocator >`, 1711
  - `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 1716
  - `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 1722
  - `std::__debug::multiset< _Key, _Compare, _Allocator >`, 1727
  - `std::__debug::set< _Key, _Compare, _Allocator >`, 1732
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1738
  - `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1744
  - `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 1750
  - `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 1756
  - `std::__debug::vector< _Tp, _Allocator >`, 1762
- `_M_detach_single`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1121
  - `__gnu_debug::Safe_iterator_base`, 1131
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1139
  - `__gnu_debug::Safe_local_iterator_base`, 1148
- `_M_detach_singular`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1112
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1151
- `__gnu_debug::Safe_sequence< _Sequence >`, 1155
- `__gnu_debug::Safe_sequence_base`, 1160
- `__gnu_debug::Safe_unordered_container< _Container >`, 1163
- `__gnu_debug::Safe_unordered_container_base`, 1168
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1176
- `std::__debug::deque< _Tp, _Allocator >`, 1701
- `std::__debug::forward_list< _Tp, _Alloc >`, 1706
- `std::__debug::list< _Tp, _Allocator >`, 1711
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 1717
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 1722
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 1727
- `std::__debug::set< _Key, _Compare, _Allocator >`, 1732
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1738
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1744
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 1750
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 1756
- `std::__debug::vector< _Tp, _Allocator >`, 1762
- `_M_dist`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 1245
- `_M_elements_leftover`
  - `__gnu_parallel::QSBThreadLocal< _RAIter >`, 1299
- `_M_end`
  - `__gnu_parallel::Piece< _DifferenceTp >`, 1290
- `_M_ext_buf`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1003
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1067
  - `std::basic_filebuf< _CharT, _Traits >`, 1988
- `_M_ext_buf_size`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1003
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1067
  - `std::basic_filebuf< _CharT, _Traits >`, 1988
- `_M_ext_next`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1004
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1068
  - `std::basic_filebuf< _CharT, _Traits >`, 1989
- `_M_fill_initialize`
  - `std::deque< _Tp, _Alloc >`, 2860
- `_M_finish_iterator`
  - `__gnu_parallel::__accumulate_selector< _It >`, 1203

- `__gnu_parallel::__adjacent_difference_selector< _It >`, 1204
- `__gnu_parallel::__count_if_selector< _It, _Diff >`, 1211
- `__gnu_parallel::__count_selector< _It, _Diff >`, 1212
- `__gnu_parallel::__fill_selector< _It >`, 1214
- `__gnu_parallel::__for_each_selector< _It >`, 1219
- `__gnu_parallel::__generate_selector< _It >`, 1221
- `__gnu_parallel::__generic_for_each_selector< _It >`, 1224
- `__gnu_parallel::__identity_selector< _It >`, 1226
- `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`, 1228
- `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`, 1236
- `__gnu_parallel::__replace_selector< _It, _Tp >`, 1239
- `__gnu_parallel::__transform1_selector< _It >`, 1240
- `__gnu_parallel::__transform2_selector< _It >`, 1242
- `_M_first`
  - `__gnu_parallel::__Job< _DifferenceTp >`, 1259
- `_M_first_insert`
  - `__gnu_parallel::__LoserTreeBase< _Tp, _Compare >`, 1274
- `_M_gcount`
  - `std::basic_fstream< _CharT, _Traits >`, 2052
  - `std::basic_ifstream< _CharT, _Traits >`, 2106
  - `std::basic_iostream< _CharT, _Traits >`, 2203
  - `std::basic_istream< _CharT, _Traits >`, 2255
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2311
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2633
- `_M_get`
  - `__gnu_cxx::free_list`, 1011
- `_M_get_mutex`
  - `__gnu_debug::__Safe_forward_list< _SafeSequence >`, 1113
  - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence >`, 1121
  - `__gnu_debug::__Safe_iterator_base`, 1131
  - `__gnu_debug::__Safe_local_iterator< _Iterator, _Sequence >`, 1139
  - `__gnu_debug::__Safe_local_iterator_base`, 1148
  - `__gnu_debug::__Safe_node_sequence< _Sequence >`, 1151
  - `__gnu_debug::__Safe_sequence< _Sequence >`, 1155
  - `__gnu_debug::__Safe_sequence_base`, 1160
  - `__gnu_debug::__Safe_unordered_container< _Container >`, 1163
  - `__gnu_debug::__Safe_unordered_container_base`, 1168
  - `__gnu_debug::__basic_string< _CharT, _Traits, _Allocator >`, 1176
- `std::__debug::deque< _Tp, _Allocator >`, 1701
- `std::__debug::forward_list< _Tp, _Alloc >`, 1706
- `std::__debug::list< _Tp, _Allocator >`, 1711
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 1717
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 1722
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 1727
- `std::__debug::set< _Key, _Compare, _Allocator >`, 1733
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1738
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1744
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 1750
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 1756
- `std::__debug::vector< _Tp, _Allocator >`, 1762
- `_M_get_result`
  - `std::__basic_future< _Res >`, 1674
  - `std::future< _Res >`, 2985
  - `std::future< _Res & >`, 2988
  - `std::future< void >`, 2991
  - `std::shared_future< _Res >`, 3619
  - `std::shared_future< _Res & >`, 3623
  - `std::shared_future< void >`, 3626
- `_M_getloc`
  - `std::basic_fstream< _CharT, _Traits >`, 2004
  - `std::basic_ifstream< _CharT, _Traits >`, 2070
  - `std::basic_ios< _CharT, _Traits >`, 2123
  - `std::basic_iostream< _CharT, _Traits >`, 2156
  - `std::basic_istream< _CharT, _Traits >`, 2220
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2275
  - `std::basic_ofstream< _CharT, _Traits >`, 2328
  - `std::basic_ostream< _CharT, _Traits >`, 2371
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 2416
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2586
  - `std::ios_base`, 3062
- `_M_global`
  - `__gnu_parallel::__QSBThreadLocal< _RAIter >`, 1299
- `_M_impl`
  - `std::deque< _Tp, _Alloc >`, 2882
- `_M_in_beg`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1004
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1068
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1092

- std::basic\_filebuf< \_CharT, \_Traits >, 1989
- std::basic\_streambuf< \_CharT, \_Traits >, 2485
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2573
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3955
- \_M\_in\_cur
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 1004
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1068
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1092
  - std::basic\_filebuf< \_CharT, \_Traits >, 1989
  - std::basic\_streambuf< \_CharT, \_Traits >, 2485
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2573
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3955
- \_M\_in\_end
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 1004
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1068
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1092
  - std::basic\_filebuf< \_CharT, \_Traits >, 1989
  - std::basic\_streambuf< \_CharT, \_Traits >, 2485
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2573
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3955
- \_M\_in\_same\_bucket
  - \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >, 1139
- \_M\_incrementable
  - \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >, 1121
  - \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >, 1139
- \_M\_initial
  - \_\_gnu\_parallel::QSBThreadLocal< \_RAIter >, 1299
- \_M\_initialize\_map
  - std::\_Deque\_base< \_Tp, \_Alloc >, 1859
  - std::deque< \_Tp, \_Alloc >, 2861
- \_M\_insert
  - \_\_gnu\_cxx::free\_list, 1011
- \_M\_invalidate
  - \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >, 1121
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 1131
  - \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >, 1140
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 1148
- \_M\_invalidate\_all
  - \_\_gnu\_debug::\_Safe\_forward\_list< \_SafeSequence >, 1113
  - \_\_gnu\_debug::\_Safe\_node\_sequence< \_Sequence >, 1152
  - \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >, 1155
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 1160
- \_\_gnu\_debug::\_Safe\_unordered\_container< \_Container >, 1163
- \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 1168
- \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1176
- std::\_debug::deque< \_Tp, \_Allocator >, 1702
- std::\_debug::forward\_list< \_Tp, \_Alloc >, 1707
- std::\_debug::list< \_Tp, \_Allocator >, 1712
- std::\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >, 1717
- std::\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, 1722
- std::\_debug::multiset< \_Key, \_Compare, \_Allocator >, 1728
- std::\_debug::set< \_Key, \_Compare, \_Allocator >, 1733
- std::\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1738
- std::\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1744
- std::\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 1750
- std::\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 1756
- std::\_debug::vector< \_Tp, \_Allocator >, 1762
- \_M\_invalidate\_if
  - \_\_gnu\_debug::\_Safe\_forward\_list< \_SafeSequence >, 1113
  - \_\_gnu\_debug::\_Safe\_node\_sequence< \_Sequence >, 1152
  - \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >, 1156
  - \_\_gnu\_debug::\_Safe\_unordered\_container< \_Container >, 1164
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1177
  - std::\_debug::deque< \_Tp, \_Allocator >, 1702
  - std::\_debug::forward\_list< \_Tp, \_Alloc >, 1707
  - std::\_debug::list< \_Tp, \_Allocator >, 1712
  - std::\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >, 1717
  - std::\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, 1723
  - std::\_debug::multiset< \_Key, \_Compare, \_Allocator >, 1728
  - std::\_debug::set< \_Key, \_Compare, \_Allocator >, 1733
  - std::\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1739
  - std::\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1745
  - std::\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 1750



- std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [1756](#)
- std::\_\_debug::vector< \_Tp, \_Allocator >, [1763](#)
- \_M\_invalidate\_local\_if
  - \_\_gnu\_debug::Safe\_unordered\_container< \_Container >, [1164](#)
  - std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [1739](#)
  - std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [1745](#)
  - std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [1751](#)
  - std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [1757](#)
- \_M\_is\_before\_begin
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1122](#)
- \_M\_is\_begin
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1122](#)
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, [1140](#)
- \_M\_is\_beginnest
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1122](#)
- \_M\_is\_end
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1122](#)
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, [1140](#)
- \_M\_iterators
  - \_\_gnu\_debug::Safe\_forward\_list< \_SafeSequence >, [1114](#)
  - \_\_gnu\_debug::Safe\_node\_sequence< \_Sequence >, [1153](#)
  - \_\_gnu\_debug::Safe\_sequence< \_Sequence >, [1157](#)
  - \_\_gnu\_debug::Safe\_sequence\_base, [1161](#)
  - \_\_gnu\_debug::Safe\_unordered\_container< \_Container >, [1165](#)
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, [1169](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1200](#)
  - std::\_\_debug::deque< \_Tp, \_Allocator >, [1703](#)
  - std::\_\_debug::forward\_list< \_Tp, \_Alloc >, [1708](#)
  - std::\_\_debug::list< \_Tp, \_Allocator >, [1713](#)
  - std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >, [1718](#)
  - std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, [1724](#)
  - std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >, [1729](#)
  - std::\_\_debug::set< \_Key, \_Compare, \_Allocator >, [1734](#)
  - std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [1740](#)
  - std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [1746](#)
  - std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [1752](#)
  - std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [1758](#)
  - std::\_\_debug::vector< \_Tp, \_Allocator >, [1764](#)
- \_M\_key
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >::Loser, [1275](#)
- \_M\_last
  - \_\_gnu\_parallel::Job< \_DifferenceTp >, [1260](#)
- \_M\_leftover\_parts
  - \_\_gnu\_parallel::QSBThreadLocal< \_RAIter >, [1299](#)
- \_M\_load
  - \_\_gnu\_parallel::Job< \_DifferenceTp >, [1260](#)
- \_M\_local\_iterators
  - \_\_gnu\_debug::Safe\_unordered\_container< \_Container >, [1165](#)
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, [1169](#)
  - std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [1740](#)
  - std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [1746](#)
  - std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [1752](#)
  - std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [1758](#)
- \_M\_log\_k
  - \_\_gnu\_parallel::LoserTree< \_\_stable, \_Tp, \_Compare >, [1268](#)
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >, [1274](#)
- \_M\_losers
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >, [1274](#)
- \_M\_mode
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [1004](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1068](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1989](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2573](#)
- \_M\_new\_elements\_at\_back
  - std::deque< \_Tp, \_Alloc >, [2862](#)
- \_M\_new\_elements\_at\_front
  - std::deque< \_Tp, \_Alloc >, [2862](#)
- \_M\_next
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1126](#)
  - \_\_gnu\_debug::Safe\_iterator\_base, [1132](#)

- `__gnu_debug::__Safe_local_iterator< _Iterator, _Sequence >`, 1144
- `__gnu_debug::__Safe_local_iterator_base`, 1149
- `_M_num_bins`
  - `__gnu_parallel::__DRandomShufflingGlobalData< _RAIter >`, 1245
- `_M_num_bits`
  - `__gnu_parallel::__DRandomShufflingGlobalData< _RAIter >`, 1246
- `_M_num_threads`
  - `__gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 1248
  - `__gnu_parallel::__PMWMSortingData< _RAIter >`, 1293
  - `__gnu_parallel::__QSBThreadLocal< _RAIter >`, 1300
- `_M_offsets`
  - `__gnu_parallel::__PMWMSortingData< _RAIter >`, 1293
- `_M_out_beg`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1005
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1069
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1092
  - `std::basic_filebuf< _CharT, _Traits >`, 1990
  - `std::basic_streambuf< _CharT, _Traits >`, 2485
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2574
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3955
- `_M_out_cur`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1005
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1069
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1092
  - `std::basic_filebuf< _CharT, _Traits >`, 1990
  - `std::basic_streambuf< _CharT, _Traits >`, 2485
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2574
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3955
- `_M_out_end`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1005
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1069
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1093
  - `std::basic_filebuf< _CharT, _Traits >`, 1990
  - `std::basic_streambuf< _CharT, _Traits >`, 2486
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2574
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3956
- `_M_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1005
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1069
  - `std::basic_filebuf< _CharT, _Traits >`, 1990
- `_M_pback_cur_save`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1005
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1070
  - `std::basic_filebuf< _CharT, _Traits >`, 1991
- `_M_pback_end_save`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1006
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1070
  - `std::basic_filebuf< _CharT, _Traits >`, 1991
- `_M_pback_init`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1006
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1070
  - `std::basic_filebuf< _CharT, _Traits >`, 1991
- `_M_pieces`
  - `__gnu_parallel::__PMWMSortingData< _RAIter >`, 1293
- `_M_pop_back_aux`
  - `std::deque< _Tp, _Alloc >`, 2862
- `_M_pop_front_aux`
  - `std::deque< _Tp, _Alloc >`, 2862
- `_M_prior`
  - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence >`, 1127
  - `__gnu_debug::__Safe_iterator_base`, 1132
  - `__gnu_debug::__Safe_local_iterator< _Iterator, _Sequence >`, 1144
  - `__gnu_debug::__Safe_local_iterator_base`, 1149
- `_M_push_back_aux`
  - `std::deque< _Tp, _Alloc >`, 2862
- `_M_push_front_aux`
  - `std::deque< _Tp, _Alloc >`, 2863
- `_M_range_check`
  - `std::deque< _Tp, _Alloc >`, 2863
  - `std::vector< _Tp, _Alloc >`, 3911
- `_M_range_initialize`
  - `std::deque< _Tp, _Alloc >`, 2863, 2864
- `_M_reading`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 1006
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1071
  - `std::basic_filebuf< _CharT, _Traits >`, 1992
- `_M_reallocate_map`
  - `std::deque< _Tp, _Alloc >`, 2864
- `_M_reserve_elements_at_back`
  - `std::deque< _Tp, _Alloc >`, 2865
- `_M_reserve_elements_at_front`
  - `std::deque< _Tp, _Alloc >`, 2865
- `_M_reserve_map_at_back`
  - `std::deque< _Tp, _Alloc >`, 2865
- `_M_reserve_map_at_front`
  - `std::deque< _Tp, _Alloc >`, 2865
- `_M_reset`
  - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence >`, 1123
  - `__gnu_debug::__Safe_iterator_base`, 1131
  - `__gnu_debug::__Safe_local_iterator< _Iterator, _Sequence >`, 1140
  - `__gnu_debug::__Safe_local_iterator_base`, 1148
- `_M_revalidate_singular`

- `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1113
- `__gnu_debug::Safe_node_sequence< _Sequence >`, 1152
- `__gnu_debug::Safe_sequence< _Sequence >`, 1156
- `__gnu_debug::Safe_sequence_base`, 1160
- `__gnu_debug::Safe_unordered_container< _Container >`, 1164
- `__gnu_debug::Safe_unordered_container_base`, 1168
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1177
- `std::__debug::deque< _Tp, _Allocator >`, 1702
- `std::__debug::forward_list< _Tp, _Alloc >`, 1707
- `std::__debug::list< _Tp, _Allocator >`, 1712
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 1718
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 1723
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 1728
- `std::__debug::set< _Key, _Compare, _Allocator >`, 1733
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1739
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1745
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 1751
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 1757
- `std::__debug::vector< _Tp, _Allocator >`, 1763
- `_M_samples`
  - `__gnu_parallel::PMWMSortingData< _RAIter >`, 1294
- `_M_sd`
  - `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 1248
- `_M_seed`
  - `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 1248
- `_M_sequence`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1127
  - `__gnu_debug::Safe_iterator_base`, 1132
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1144
  - `__gnu_debug::Safe_local_iterator_base`, 1149
- `_M_sequential_algorithm`
  - `__gnu_parallel::__adjacent_find_selector`, 1205
  - `__gnu_parallel::__find_first_of_selector< _FIterator >`, 1215
  - `__gnu_parallel::__find_if_selector`, 1217
  - `__gnu_parallel::__mismatch_selector`, 1230
- `_M_set_buffer`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 984
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1047
  - `std::basic_filebuf< _CharT, _Traits >`, 1968
- `_M_set_node`
  - `std::Deque_iterator< _Tp, _Ref, _Ptr >`, 1861
- `_M_singular`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1123
  - `__gnu_debug::Safe_iterator_base`, 1131
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1141
  - `__gnu_debug::Safe_local_iterator_base`, 1148
- `_M_source`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 1246
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::_Loser`, 1275
  - `__gnu_parallel::PMWMSortingData< _RAIter >`, 1294
- `_M_starts`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 1246
  - `__gnu_parallel::PMWMSortingData< _RAIter >`, 1294
- `_M_sup`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::_Loser`, 1276
- `_M_swap`
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1152
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1156
  - `__gnu_debug::Safe_sequence_base`, 1160
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1164
  - `__gnu_debug::Safe_unordered_container_base`, 1168, 1169
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1177
  - `std::__debug::deque< _Tp, _Allocator >`, 1702
  - `std::__debug::list< _Tp, _Allocator >`, 1712
  - `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 1718
  - `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 1723
  - `std::__debug::multiset< _Key, _Compare, _Allocator >`, 1728
  - `std::__debug::set< _Key, _Compare, _Allocator >`, 1734
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 1739
  - `std::__debug::unordered_multimap< _Key, _Tp,`

- [\\_Hash, \\_Pred, \\_Alloc >, 1745](#)
- [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 1751](#)
- [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 1757](#)
- [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 1763](#)
- [\\_M\\_temporaries](#)
  - [\\_\\_gnu\\_parallel::\\_\\_DRandomShufflingGlobalData< \\_RAIter >, 1246](#)
- [\\_M\\_temporary](#)
  - [\\_\\_gnu\\_parallel::\\_\\_PMWMSortingData< \\_RAIter >, 1294](#)
- [\\_M\\_transfer\\_from\\_if](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_forward\\_list< \\_SafeSequence >, 1114](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_node\\_sequence< \\_Sequence >, 1152](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence< \\_Sequence >, 1156](#)
  - [\\_\\_gnu\\_debug::\\_\\_basic\\_string< \\_CharT, \\_Traits, \\_Allocator >, 1177](#)
  - [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >, 1702](#)
  - [std::\\_\\_debug::forward\\_list< \\_Tp, \\_Alloc >, 1707](#)
  - [std::\\_\\_debug::list< \\_Tp, \\_Allocator >, 1712](#)
  - [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 1718](#)
  - [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 1723](#)
  - [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >, 1728](#)
  - [std::\\_\\_debug::set< \\_Key, \\_Compare, \\_Allocator >, 1734](#)
  - [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 1763](#)
- [\\_M\\_unlink](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator< \\_Iterator, \\_Sequence >, 1123](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator\\_base, 1131](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator< \\_Iterator, \\_Sequence >, 1141](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator\\_base, 1148](#)
- [\\_M\\_use\\_pointer](#)
  - [\\_\\_gnu\\_parallel::\\_\\_LoserTreeTraits< \\_Tp >, 1284](#)
- [\\_M\\_version](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_forward\\_list< \\_SafeSequence >, 1114](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator< \\_Iterator, \\_Sequence >, 1127](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator\\_base, 1132](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator< \\_Iterator, \\_Sequence >, 1144](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator\\_base, 1149](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_node\\_sequence< \\_Sequence >, 1153](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence< \\_Sequence >, 1157](#)
- [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence\\_base, 1161](#)
- [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container< \\_Container >, 1166](#)
- [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container\\_base, 1170](#)
- [\\_\\_gnu\\_debug::\\_\\_basic\\_string< \\_CharT, \\_Traits, \\_Allocator >, 1200](#)
- [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >, 1703](#)
- [std::\\_\\_debug::forward\\_list< \\_Tp, \\_Alloc >, 1708](#)
- [std::\\_\\_debug::list< \\_Tp, \\_Allocator >, 1713](#)
- [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 1719](#)
- [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 1724](#)
- [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >, 1729](#)
- [std::\\_\\_debug::set< \\_Key, \\_Compare, \\_Allocator >, 1734](#)
- [std::\\_\\_debug::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 1741](#)
- [std::\\_\\_debug::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 1747](#)
- [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 1752](#)
- [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 1758](#)
- [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 1764](#)
- [\\_M\\_w](#)
  - [std::\\_\\_Base\\_bitset< \\_Nw >, 1854](#)
  - [std::\\_\\_tr2::\\_\\_dynamic\\_bitset\\_base< \\_WordT, \\_Alloc >, 3708](#)
- [\\_M\\_write](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >, 2004](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >, 2156](#)
  - [std::basic\\_ofstream< \\_CharT, \\_Traits >, 2328](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >, 2371](#)
  - [std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >, 2416](#)
  - [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >, 2587](#)
- [\\_MultiwayMergeAlgorithm](#)
  - [\\_\\_gnu\\_parallel, 504](#)
- [\\_Opcode](#)
  - [Base and Implementation Classes, 30](#)
- [\\_PCCFP](#)
  - [\\_\\_gnu\\_parallel::\\_\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >, 1257](#)
  - [std::pair< \\_T1, \\_T2 >, 3504](#)
  - [std::sub\\_match< \\_Biliter >, 3656](#)
- [\\_PCCP](#)
  - [\\_\\_gnu\\_parallel::\\_\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >, 1257](#)
  - [std::pair< \\_T1, \\_T2 >, 3504](#)

- std::sub\_match< \_Bilter >, 3656
- \_Parallelism
  - \_\_gnu\_parallel, 504
- \_PartialSumAlgorithm
  - \_\_gnu\_parallel, 504
- \_Piece
  - \_\_gnu\_parallel::QSBThreadLocal< \_RAIter >, 1298
- \_PseudoSequence
  - \_\_gnu\_parallel::PseudoSequence< \_Tp, \_DifferenceTp >, 1296
- \_Ptr
  - std::\_\_basic\_future< \_Res >, 1674
  - std::\_\_future\_base, 1813
  - std::future< \_Res >, 2985
  - std::future< \_Res & >, 2987
  - std::future< void >, 2990
  - std::shared\_future< \_Res >, 3618
  - std::shared\_future< \_Res & >, 3622
  - std::shared\_future< void >, 3625
- \_QSBThreadLocal
  - \_\_gnu\_parallel::QSBThreadLocal< \_RAIter >, 1298
- \_RandomNumber
  - \_\_gnu\_parallel::RandomNumber, 1300, 1301
- \_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, 1303
- \_Safe\_iterator
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, 1117, 1118
- \_Safe\_iterator\_base
  - \_\_gnu\_debug::Safe\_iterator\_base, 1129
- \_Safe\_local\_iterator
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, 1135, 1136
- \_Safe\_local\_iterator\_base
  - \_\_gnu\_debug::Safe\_local\_iterator\_base, 1146
- \_SequenceIndex
  - \_\_gnu\_parallel, 503
- \_SortAlgorithm
  - \_\_gnu\_parallel, 505
- \_SplittingAlgorithm
  - \_\_gnu\_parallel, 505
- \_Temporary\_buffer
  - std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1891
- \_ThreadIndex
  - \_\_gnu\_parallel, 503
- \_TokenT
  - std::\_\_detail::Scanner< \_CharT >, 1809
- \_Unchecked\_flip
  - SGL, 376
- \_Unchecked\_reset
  - SGL, 377
- \_Unchecked\_set
  - SGL, 377
- \_Unchecked\_test
  - SGL, 377
- \_\_addressof
  - Utilities, 437
- \_\_allocate\_guarded
  - std, 673
- \_\_allocated\_ptr
  - std::\_\_allocated\_ptr< \_Alloc >, 1668, 1669
- \_\_allocator\_base
  - Allocators, 10
- \_\_begin1\_iterator
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 1228
- \_\_begin2\_iterator
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 1228
- \_\_bins\_end
  - \_\_gnu\_parallel::DRSSorterPU< \_RAIter, \_RandomNumberGenerator >, 1247
- \_\_bit\_allocate
  - \_\_gnu\_cxx::\_\_detail, 480
- \_\_bit\_free
  - \_\_gnu\_cxx::\_\_detail, 480
- \_\_calc\_borders
  - \_\_gnu\_parallel, 505
- \_\_check\_dereferenceable
  - \_\_gnu\_debug, 489, 490
- \_\_check\_singular
  - \_\_gnu\_debug, 490
- \_\_check\_singular\_aux
  - \_\_gnu\_debug, 490
- \_\_check\_string
  - \_\_gnu\_debug, 490, 491
- \_\_clp2
  - Base and Implementation Classes, 33
- \_\_compare\_and\_swap
  - \_\_gnu\_parallel, 506
- \_\_constexpr\_addressof
  - optional, 4229
  - Optional values, 287
- \_\_cpp\_lib\_shared\_timed\_mutex
  - Mutexes, 207
- \_\_ctype\_type
  - std::basic\_ios< \_CharT, \_Traits >, 2117
- \_\_cxa\_demangle
  - cxxabi.h, 4078
- \_\_cxxabiv1::\_\_forced\_unwind, 864
- \_\_decode2
  - \_\_gnu\_parallel, 506
- \_\_delete\_min\_insert

- \_\_gnu\_parallel::LoserTree< \_\_stable, \_Tp, \_Compare >, [1267](#)
- \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, [1269](#)
- \_\_determine\_samples
  - \_\_gnu\_parallel, [507](#)
- \_\_encode2
  - \_\_gnu\_parallel, [507](#)
- \_\_env\_t
  - \_\_gnu\_profile, [561](#)
- \_\_equally\_split
  - \_\_gnu\_parallel, [508](#)
- \_\_equally\_split\_point
  - \_\_gnu\_parallel, [508](#)
- \_\_fetch\_and\_add
  - \_\_gnu\_parallel, [509](#)
- \_\_final\_insertion\_sort
  - std, [673](#)
- \_\_find\_if
  - std, [674](#)
- \_\_find\_if\_not
  - std, [674](#)
- \_\_find\_if\_not\_n
  - std, [675](#)
- \_\_find\_template
  - \_\_gnu\_parallel, [509–511](#)
- \_\_for\_each\_template\_random\_access
  - \_\_gnu\_parallel, [512](#)
- \_\_for\_each\_template\_random\_access\_ed
  - \_\_gnu\_parallel, [514](#)
- \_\_for\_each\_template\_random\_access\_omp\_loop
  - \_\_gnu\_parallel, [515](#)
- \_\_for\_each\_template\_random\_access\_omp\_loop\_static
  - \_\_gnu\_parallel, [515](#)
- \_\_for\_each\_template\_random\_access\_workstealing
  - \_\_gnu\_parallel, [516](#)
- \_\_foreign\_iterator\_aux2
  - \_\_gnu\_debug, [491](#)
- \_\_gcd
  - std, [675](#)
- \_\_gen\_two\_uniform\_ints
  - std, [675](#)
- \_\_genrand\_bits
  - \_\_gnu\_parallel::RandomNumber, [1301](#)
- \_\_get\_distance
  - \_\_gnu\_debug, [491](#), [492](#)
- \_\_get\_min\_source
  - \_\_gnu\_parallel::LoserTree< \_\_stable, \_Tp, \_Compare >, [1267](#)
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, [1269](#)
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >, [1273](#)
- \_\_get\_num\_threads
- \_\_gnu\_parallel::balanced\_quicksort\_tag, [1318](#)
- \_\_gnu\_parallel::balanced\_tag, [1319](#)
- \_\_gnu\_parallel::default\_parallel\_tag, [1321](#)
- \_\_gnu\_parallel::exact\_tag, [1323](#)
- \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, [1326](#)
- \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, [1327](#)
- \_\_gnu\_parallel::multiway\_mergesort\_tag, [1329](#)
- \_\_gnu\_parallel::omp\_loop\_static\_tag, [1330](#)
- \_\_gnu\_parallel::omp\_loop\_tag, [1331](#)
- \_\_gnu\_parallel::parallel\_tag, [1334](#)
- \_\_gnu\_parallel::quicksort\_tag, [1336](#)
- \_\_gnu\_parallel::sampling\_tag, [1337](#)
- \_\_gnu\_parallel::unbalanced\_tag, [1339](#)
- \_\_glibcxx\_check\_erase
  - macros.h, [4185](#)
- \_\_glibcxx\_check\_erase\_after
  - macros.h, [4185](#)
- \_\_glibcxx\_check\_erase\_range
  - macros.h, [4185](#)
- \_\_glibcxx\_check\_erase\_range\_after
  - macros.h, [4185](#)
- \_\_glibcxx\_check\_heap\_pred
  - macros.h, [4186](#)
- \_\_glibcxx\_check\_insert
  - macros.h, [4186](#)
- \_\_glibcxx\_check\_insert\_after
  - macros.h, [4186](#)
- \_\_glibcxx\_check\_insert\_range
  - macros.h, [4186](#)
- \_\_glibcxx\_check\_insert\_range\_after
  - macros.h, [4187](#)
- \_\_glibcxx\_check\_partitioned\_lower
  - macros.h, [4187](#)
- \_\_glibcxx\_check\_partitioned\_lower\_pred
  - macros.h, [4187](#)
- \_\_glibcxx\_check\_partitioned\_upper\_pred
  - macros.h, [4188](#)
- \_\_glibcxx\_check\_sorted\_pred
  - macros.h, [4188](#)
- \_\_gnu\_cxx, [447](#)
  - \_Bit\_scan\_forward, [463](#)
  - \_\_static\_pointer\_cast, [462](#)
  - airy\_ai, [463](#)
  - airy\_aif, [463](#)
  - airy\_ail, [463](#)
  - airy\_bi, [464](#)
  - airy\_bif, [464](#)
  - airy\_bil, [464](#)
  - conf\_hyperg, [464](#)
  - conf\_hypergf, [465](#)
  - conf\_hypergl, [465](#)
  - hyperg, [465](#)

- hypergf, [466](#)
- hypergl, [466](#)
- operator!=, [467](#), [468](#)
- operator<, [471](#), [472](#)
- operator<=, [472](#), [473](#)
- operator>, [476](#), [477](#)
- operator>=, [477](#), [478](#)
- operator+, [468–470](#)
- operator==, [474](#), [475](#)
- swap, [479](#)
- \_\_gnu\_cxx::\_\_Caster< \_ToType >, [960](#)
- \_\_gnu\_cxx::\_\_Char\_types< \_CharT >, [960](#)
- \_\_gnu\_cxx::\_\_ExtPtr\_allocator< \_Tp >, [961](#)
- \_\_gnu\_cxx::\_\_Invalid\_type, [962](#)
- \_\_gnu\_cxx::\_\_Pointer\_adapter< \_Storage\_policy >, [962](#)
- \_\_gnu\_cxx::\_\_Relative\_pointer\_impl< \_Tp >, [965](#)
- \_\_gnu\_cxx::\_\_Relative\_pointer\_impl< const \_Tp >, [965](#)
- \_\_gnu\_cxx::\_\_Std\_pointer\_impl< \_Tp >, [966](#)
- \_\_gnu\_cxx::\_\_Unqualified\_type< \_Tp >, [967](#)
- \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [865](#)
  - allocate, [868–870](#)
  - const\_void\_pointer, [867](#)
  - construct, [870](#), [871](#)
  - deallocate, [871](#), [872](#)
  - destroy, [872](#), [873](#)
  - is\_always\_equal, [867](#)
  - max\_size, [873](#)
  - propagate\_on\_container\_copy\_assignment, [867](#)
  - propagate\_on\_container\_move\_assignment, [867](#)
  - propagate\_on\_container\_swap, [868](#)
  - select\_on\_container\_copy\_construction, [874](#)
  - void\_pointer, [868](#)
- \_\_gnu\_cxx::\_\_common\_pool\_policy< \_PoolTp, \_Thread >, [875](#)
- \_\_gnu\_cxx::\_\_detail, [479](#)
  - \_bit\_allocate, [480](#)
  - \_bit\_free, [480](#)
  - \_num\_bitmaps, [480](#)
  - \_num\_blocks, [481](#)
- \_\_gnu\_cxx::\_\_detail::Bitmap\_counter< \_Tp >, [876](#)
- \_\_gnu\_cxx::\_\_detail::Ffit\_finder< \_Tp >, [877](#)
  - argument\_type, [877](#)
  - result\_type, [878](#)
- \_\_gnu\_cxx::\_\_detail::\_\_mini\_vector< \_Tp >, [875](#)
- \_\_gnu\_cxx::\_\_mt\_alloc< \_Tp, \_Poolp >, [878](#)
- \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp >, [879](#)
- \_\_gnu\_cxx::\_\_per\_type\_pool\_policy< \_Tp, \_PoolTp, \_Thread >, [880](#)
- \_\_gnu\_cxx::\_\_pool< \_Thread >, [881](#)
- \_\_gnu\_cxx::\_\_pool< false >, [881](#)
- \_\_gnu\_cxx::\_\_pool< true >, [882](#)
- \_\_gnu\_cxx::\_\_pool\_alloc< \_Tp >, [883](#)
- \_\_gnu\_cxx::\_\_pool\_alloc\_base, [885](#)
- \_\_gnu\_cxx::\_\_pool\_base, [886](#)
- \_\_gnu\_cxx::\_\_rc\_string\_base< \_CharT, \_Traits, \_Alloc >, [887](#)
- \_\_gnu\_cxx::\_\_scoped\_lock, [890](#)
- \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [890](#)
  - \_\_versa\_string, [894–897](#)
  - ~\_\_versa\_string, [898](#)
- append, [898–901](#)
- assign, [902–906](#)
- at, [907](#)
- back, [908](#)
- begin, [908](#), [909](#)
- c\_str, [909](#)
- capacity, [909](#)
- cbegin, [909](#)
- cend, [910](#)
- clear, [910](#)
- compare, [910–913](#)
- copy, [914](#)
- crbegin, [915](#)
- crend, [915](#)
- data, [915](#)
- empty, [916](#)
- end, [916](#)
- erase, [917](#), [918](#)
- find, [918–920](#)
- find\_first\_not\_of, [921–923](#)
- find\_first\_of, [923–925](#)
- find\_last\_not\_of, [925–927](#)
- find\_last\_of, [928](#), [929](#)
- front, [930](#)
- get\_allocator, [930](#)
- insert, [931–936](#)
- length, [937](#)
- max\_size, [937](#)
- npos, [959](#)
- operator+=, [937](#), [939](#), [940](#)
- operator=, [940–942](#)
- operator[], [942](#), [943](#)
- pop\_back, [943](#)
- push\_back, [944](#)
- rbegin, [944](#)
- rend, [945](#)
- replace, [945–953](#)
- reserve, [954](#)
- resize, [954](#), [955](#)
- rfind, [955–957](#)
- shrink\_to\_fit, [957](#)
- size, [958](#)
- substr, [958](#)
- swap, [959](#)
- \_\_gnu\_cxx::annotate\_base, [967](#)
- \_\_gnu\_cxx::array\_allocator< \_Tp, \_Array >, [968](#)
- \_\_gnu\_cxx::array\_allocator\_base< \_Tp >, [969](#)



- \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >, 970
- argument\_type, 971
- result\_type, 971
- \_\_gnu\_cxx::bitmap\_allocator< \_Tp >, 972
- \_M\_allocate\_single\_object, 973
- \_M\_deallocate\_single\_object, 974
- \_\_gnu\_cxx::char\_traits< \_CharT >, 974
- \_\_gnu\_cxx::character< \_Value, \_Int, \_St >, 976
- \_\_gnu\_cxx::condition\_base, 976
- \_\_gnu\_cxx::constant\_binary\_fun< \_Result, \_Arg1, \_Arg2 >, 977
- \_\_gnu\_cxx::constant\_unary\_fun< \_Result, \_Argument >, 978
- \_\_gnu\_cxx::constant\_void\_fun< \_Result >, 978
- \_\_gnu\_cxx::debug\_allocator< \_Alloc >, 979
- \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 980
- \_M\_buf, 1003
- \_M\_buf\_locale, 1003
- \_M\_buf\_size, 1003
- \_M\_create\_pback, 983
- \_M\_destroy\_pback, 984
- \_M\_ext\_buf, 1003
- \_M\_ext\_buf\_size, 1003
- \_M\_ext\_next, 1004
- \_M\_in\_beg, 1004
- \_M\_in\_cur, 1004
- \_M\_in\_end, 1004
- \_M\_mode, 1004
- \_M\_out\_beg, 1005
- \_M\_out\_cur, 1005
- \_M\_out\_end, 1005
- \_M\_pback, 1005
- \_M\_pback\_cur\_save, 1005
- \_M\_pback\_end\_save, 1006
- \_M\_pback\_init, 1006
- \_M\_reading, 1006
- \_M\_set\_buffer, 984
- close, 984
- eback, 984
- egptr, 985
- epptr, 985
- gbump, 985
- getloc, 986
- gptr, 986
- imbue, 986
- in\_avail, 987
- is\_open, 987
- open, 987, 988
- overflow, 989
- pbackfail, 989
- pbase, 991
- pbump, 991
- pptr, 992
- pubimbue, 992
- pubseekoff, 992
- pubseekpos, 993
- pubsetbuf, 993
- pubsync, 994
- sbumpc, 994
- seekoff, 994
- seekpos, 994
- setbuf, 995
- setg, 995
- setp, 996
- sgetc, 996
- sgetn, 997
- showmanyc, 997
- snextc, 998
- sputbackc, 998
- sputc, 999
- sputn, 999
- sungetc, 1000
- sync, 1000
- uflow, 1000
- underflow, 1001
- xsgetn, 1001
- xspn, 1002
- \_\_gnu\_cxx::encoding\_char\_traits< \_CharT >, 1007
- \_\_gnu\_cxx::encoding\_state, 1008
- \_\_gnu\_cxx::forced\_error, 1009
- \_\_gnu\_cxx::free\_list, 1010
- \_M\_clear, 1011
- \_M\_get, 1011
- \_M\_insert, 1011
- \_\_gnu\_cxx::hash\_map< \_Key, \_Tp, \_HashFn, \_EqualKey, \_Alloc >, 1012
- \_\_gnu\_cxx::hash\_multimap< \_Key, \_Tp, \_HashFn, \_EqualKey, \_Alloc >, 1014
- \_\_gnu\_cxx::hash\_multiset< \_Value, \_HashFcn, \_EqualKey, \_Alloc >, 1015
- \_\_gnu\_cxx::hash\_set< \_Value, \_HashFcn, \_EqualKey, \_Alloc >, 1017
- \_\_gnu\_cxx::limit\_condition, 1019
- \_\_gnu\_cxx::limit\_condition::always\_adjustor, 1019
- \_\_gnu\_cxx::limit\_condition::limit\_adjustor, 1020
- \_\_gnu\_cxx::limit\_condition::never\_adjustor, 1020
- \_\_gnu\_cxx::malloc\_allocator< \_Tp >, 1020
- \_\_gnu\_cxx::new\_allocator< \_Tp >, 1021
- \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, 1023
- first\_argument\_type, 1023
- result\_type, 1023
- second\_argument\_type, 1023
- \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, 1024
- first\_argument\_type, 1024
- result\_type, 1025
- second\_argument\_type, 1025
- \_\_gnu\_cxx::random\_condition, 1025



- `__gnu_cxx::random_condition::always_adjustor`, 1026
- `__gnu_cxx::random_condition::group_adjustor`, 1026
- `__gnu_cxx::random_condition::never_adjustor`, 1027
- `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`, 1027
- `__gnu_cxx::recursive_init_error`, 1031
- `__gnu_cxx::rope< _CharT, _Alloc >`, 1032
- `__gnu_cxx::select1st< _Pair >`, 1037
  - `argument_type`, 1037
  - `result_type`, 1037
- `__gnu_cxx::select2nd< _Pair >`, 1038
  - `argument_type`, 1038
  - `result_type`, 1039
- `__gnu_cxx::slist< _Tp, _Alloc >`, 1039
- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1042
  - `_M_buf`, 1067
  - `_M_buf_locale`, 1067
  - `_M_buf_size`, 1067
  - `_M_create_pback`, 1046
  - `_M_destroy_pback`, 1047
  - `_M_ext_buf`, 1067
  - `_M_ext_buf_size`, 1067
  - `_M_ext_next`, 1068
  - `_M_in_beg`, 1068
  - `_M_in_cur`, 1068
  - `_M_in_end`, 1068
  - `_M_mode`, 1068
  - `_M_out_beg`, 1069
  - `_M_out_cur`, 1069
  - `_M_out_end`, 1069
  - `_M_pback`, 1069
  - `_M_pback_cur_save`, 1070
  - `_M_pback_end_save`, 1070
  - `_M_pback_init`, 1070
  - `_M_reading`, 1071
  - `_M_set_buffer`, 1047
  - `~stdio_filebuf`, 1046
  - `close`, 1047
  - `eback`, 1048
  - `egptr`, 1048
  - `epptr`, 1048
  - `fd`, 1049
  - `file`, 1049
  - `gbump`, 1049
  - `getloc`, 1050
  - `gptr`, 1050
  - `imbue`, 1050
  - `in_avail`, 1051
  - `is_open`, 1051
  - `open`, 1051, 1052
  - `overflow`, 1053
  - `pbackfail`, 1053
  - `pbase`, 1055
  - `pbump`, 1055
  - `pptr`, 1056
  - `pubimbue`, 1056
  - `pubseekoff`, 1056
  - `pubseekpos`, 1057
  - `pubsetbuf`, 1057
  - `pubsync`, 1058
  - `sbumpc`, 1058
  - `seekoff`, 1058
  - `seekpos`, 1058
  - `setbuf`, 1059
  - `setg`, 1059
  - `setp`, 1060
  - `sgetc`, 1060
  - `sgetn`, 1061
  - `showmanyc`, 1061
  - `snextc`, 1062
  - `sputbackc`, 1062
  - `sputc`, 1063
  - `sputn`, 1063
  - `stdio_filebuf`, 1045, 1046
  - `sungetc`, 1064
  - `sync`, 1064
  - `uflow`, 1064
  - `underflow`, 1065
  - `xsggetn`, 1065
  - `xspn`, 1066
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1071
  - `_M_buf_locale`, 1091
  - `_M_in_beg`, 1092
  - `_M_in_cur`, 1092
  - `_M_in_end`, 1092
  - `_M_out_beg`, 1092
  - `_M_out_cur`, 1092
  - `_M_out_end`, 1093
  - `eback`, 1074
  - `egptr`, 1074
  - `epptr`, 1075
  - `file`, 1075
  - `gbump`, 1075
  - `getloc`, 1076
  - `gptr`, 1076
  - `imbue`, 1076
  - `in_avail`, 1077
  - `overflow`, 1077
  - `pbackfail`, 1078
  - `pbase`, 1078
  - `pbump`, 1079
  - `pptr`, 1079
  - `pubimbue`, 1079
  - `pubseekoff`, 1080
  - `pubseekpos`, 1080
  - `pubsetbuf`, 1082
  - `pubsync`, 1082
  - `sbumpc`, 1082

- seekoff, 1082
- seekpos, 1083
- setbuf, 1083
- setg, 1084
- setp, 1084
- sgetc, 1085
- sgetn, 1085
- showmanyc, 1086
- snextc, 1086
- sputbackc, 1087
- sputc, 1087
- sputn, 1088
- sungetc, 1088
- sync, 1089
- uflow, 1089
- underflow, 1089
- xsgetn, 1090
- xspn, 1091
- \_\_gnu\_cxx::subtractive\_rng, 1093
  - argument\_type, 1094
  - operator(), 1095
  - result\_type, 1094
  - subtractive\_rng, 1094
- \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 1095
  - ~temporary\_buffer, 1097
  - begin, 1097
  - end, 1097
  - requested\_size, 1097
  - size, 1098
  - temporary\_buffer, 1096
- \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >, 1098
- \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >, 1100
- \_\_gnu\_cxx::throw\_allocator\_random< \_Tp >, 1102
- \_\_gnu\_cxx::throw\_value\_base< \_Cond >, 1103
- \_\_gnu\_cxx::throw\_value\_limit, 1105
- \_\_gnu\_cxx::throw\_value\_random, 1106
- \_\_gnu\_cxx::typelist, 481
  - apply\_generator, 482
- \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >, 1107
  - argument\_type, 1108
  - result\_type, 1108
- \_\_gnu\_debug, 482
  - \_Distance\_precision, 489
  - \_\_check\_dereferenceable, 489, 490
  - \_\_check\_singular, 490
  - \_\_check\_singular\_aux, 490
  - \_\_check\_string, 490, 491
  - \_\_foreign\_iterator\_aux2, 491
  - \_\_get\_distance, 491, 492
  - \_\_valid\_range, 492, 493
  - \_\_valid\_range\_aux, 493, 494
- \_\_gnu\_debug::After\_nth\_from< \_Iterator >, 1109
- \_\_gnu\_debug::BeforeBeginHelper< \_Sequence >, 1109
- \_\_gnu\_debug::Equal\_to< \_Type >, 1110
- \_\_gnu\_debug::Not\_equal\_to< \_Type >, 1110
- \_\_gnu\_debug::Safe\_container< \_SafeContainer, \_Alloc, \_SafeBase, \_IsCxx11AllocatorAware >, 1110
- \_\_gnu\_debug::Safe\_forward\_list< \_SafeSequence >, 1111
  - \_M\_const\_iterators, 1114
  - \_M\_detach\_all, 1112
  - \_M\_detach\_singular, 1112
  - \_M\_get\_mutex, 1113
  - \_M\_invalidate\_all, 1113
  - \_M\_invalidate\_if, 1113
  - \_M\_iterators, 1114
  - \_M\_revalidate\_singular, 1113
  - \_M\_transfer\_from\_if, 1114
  - \_M\_version, 1114
- \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, 1115
  - \_M\_attach, 1119
  - \_M\_attach\_single, 1119, 1120
  - \_M\_attached\_to, 1120
  - \_M\_before\_dereferenceable, 1120
  - \_M\_can\_compare, 1120
  - \_M\_dereferenceable, 1121
  - \_M\_detach, 1121
  - \_M\_detach\_single, 1121
  - \_M\_get\_mutex, 1121
  - \_M\_incrementable, 1121
  - \_M\_invalidate, 1121
  - \_M\_is\_before\_begin, 1122
  - \_M\_is\_begin, 1122
  - \_M\_is\_beginnest, 1122
  - \_M\_is\_end, 1122
  - \_M\_next, 1126
  - \_M\_prior, 1127
  - \_M\_reset, 1123
  - \_M\_sequence, 1127
  - \_M\_singular, 1123
  - \_M\_unlink, 1123
  - \_M\_version, 1127
  - \_Safe\_iterator, 1117, 1118
  - base, 1123
  - operator \_Iterator, 1124
  - operator\*, 1124
  - operator++, 1124
  - operator->, 1125
  - operator--, 1125
  - operator=, 1126
- \_\_gnu\_debug::Safe\_iterator\_base, 1128
  - \_M\_attach, 1130
  - \_M\_attach\_single, 1130
  - \_M\_attached\_to, 1130
  - \_M\_can\_compare, 1130

- [\\_M\\_detach, 1130](#)
- [\\_M\\_detach\\_single, 1131](#)
- [\\_M\\_get\\_mutex, 1131](#)
- [\\_M\\_invalidate, 1131](#)
- [\\_M\\_next, 1132](#)
- [\\_M\\_prior, 1132](#)
- [\\_M\\_reset, 1131](#)
- [\\_M\\_sequence, 1132](#)
- [\\_M\\_singular, 1131](#)
- [\\_M\\_unlink, 1131](#)
- [\\_M\\_version, 1132](#)
- [\\_Safe\\_iterator\\_base, 1129](#)
- [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator< \\_Iterator, \\_Sequence >, 1133](#)
- [\\_M\\_attach, 1137](#)
- [\\_M\\_attach\\_single, 1137, 1138](#)
- [\\_M\\_attached\\_to, 1138](#)
- [\\_M\\_can\\_compare, 1138](#)
- [\\_M\\_dereferenceable, 1138](#)
- [\\_M\\_detach, 1139](#)
- [\\_M\\_detach\\_single, 1139](#)
- [\\_M\\_get\\_mutex, 1139](#)
- [\\_M\\_in\\_same\\_bucket, 1139](#)
- [\\_M\\_incrementable, 1139](#)
- [\\_M\\_invalidate, 1140](#)
- [\\_M\\_is\\_begin, 1140](#)
- [\\_M\\_is\\_end, 1140](#)
- [\\_M\\_next, 1144](#)
- [\\_M\\_prior, 1144](#)
- [\\_M\\_reset, 1140](#)
- [\\_M\\_sequence, 1144](#)
- [\\_M\\_singular, 1141](#)
- [\\_M\\_unlink, 1141](#)
- [\\_M\\_version, 1144](#)
- [\\_Safe\\_local\\_iterator, 1135, 1136](#)
- [base, 1141](#)
- [bucket, 1141](#)
- [operator \\_iterator, 1142](#)
- [operator\\*, 1142](#)
- [operator++, 1142](#)
- [operator->, 1143](#)
- [operator=, 1143](#)
- [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator\\_base, 1145](#)
- [\\_M\\_attach, 1147](#)
- [\\_M\\_attach\\_single, 1147](#)
- [\\_M\\_attached\\_to, 1147](#)
- [\\_M\\_can\\_compare, 1147](#)
- [\\_M\\_detach, 1147](#)
- [\\_M\\_detach\\_single, 1148](#)
- [\\_M\\_get\\_mutex, 1148](#)
- [\\_M\\_invalidate, 1148](#)
- [\\_M\\_next, 1149](#)
- [\\_M\\_prior, 1149](#)
- [\\_M\\_reset, 1148](#)
- [\\_M\\_sequence, 1149](#)
- [\\_M\\_singular, 1148](#)
- [\\_M\\_unlink, 1148](#)
- [\\_M\\_version, 1149](#)
- [\\_Safe\\_local\\_iterator\\_base, 1146](#)
- [\\_\\_gnu\\_debug:: Safe\\_node\\_sequence< \\_Sequence >, 1150](#)
- [\\_M\\_const\\_iterators, 1153](#)
- [\\_M\\_detach\\_all, 1151](#)
- [\\_M\\_detach\\_singular, 1151](#)
- [\\_M\\_get\\_mutex, 1151](#)
- [\\_M\\_invalidate\\_all, 1152](#)
- [\\_M\\_invalidate\\_if, 1152](#)
- [\\_M\\_iterators, 1153](#)
- [\\_M\\_revalidate\\_singular, 1152](#)
- [\\_M\\_swap, 1152](#)
- [\\_M\\_transfer\\_from\\_if, 1152](#)
- [\\_M\\_version, 1153](#)
- [\\_\\_gnu\\_debug:: Safe\\_sequence< \\_Sequence >, 1154](#)
- [\\_M\\_const\\_iterators, 1157](#)
- [\\_M\\_detach\\_all, 1155](#)
- [\\_M\\_detach\\_singular, 1155](#)
- [\\_M\\_get\\_mutex, 1155](#)
- [\\_M\\_invalidate\\_all, 1155](#)
- [\\_M\\_invalidate\\_if, 1156](#)
- [\\_M\\_iterators, 1157](#)
- [\\_M\\_revalidate\\_singular, 1156](#)
- [\\_M\\_swap, 1156](#)
- [\\_M\\_transfer\\_from\\_if, 1156](#)
- [\\_M\\_version, 1157](#)
- [\\_\\_gnu\\_debug:: Safe\\_sequence\\_base, 1158](#)
- [\\_M\\_const\\_iterators, 1161](#)
- [\\_M\\_detach\\_all, 1159](#)
- [\\_M\\_detach\\_singular, 1160](#)
- [\\_M\\_get\\_mutex, 1160](#)
- [\\_M\\_invalidate\\_all, 1160](#)
- [\\_M\\_iterators, 1161](#)
- [\\_M\\_revalidate\\_singular, 1160](#)
- [\\_M\\_swap, 1160](#)
- [\\_M\\_version, 1161](#)
- [~\\_Safe\\_sequence\\_base, 1159](#)
- [\\_\\_gnu\\_debug:: Safe\\_unordered\\_container< \\_Container >, 1162](#)
- [\\_M\\_const\\_iterators, 1165](#)
- [\\_M\\_const\\_local\\_iterators, 1165](#)
- [\\_M\\_detach\\_all, 1163](#)
- [\\_M\\_detach\\_singular, 1163](#)
- [\\_M\\_get\\_mutex, 1163](#)
- [\\_M\\_invalidate\\_all, 1163](#)
- [\\_M\\_invalidate\\_if, 1164](#)
- [\\_M\\_invalidate\\_local\\_if, 1164](#)
- [\\_M\\_iterators, 1165](#)
- [\\_M\\_local\\_iterators, 1165](#)
- [\\_M\\_revalidate\\_singular, 1164](#)

- [\\_M\\_swap, 1164](#)
- [\\_M\\_version, 1166](#)
- [\\_\\_gnu\\_debug:: Safe\\_unordered\\_container\\_base, 1166](#)
  - [\\_M\\_const\\_iterators, 1169](#)
  - [\\_M\\_const\\_local\\_iterators, 1169](#)
  - [\\_M\\_detach\\_all, 1167](#)
  - [\\_M\\_detach\\_singular, 1168](#)
  - [\\_M\\_get\\_mutex, 1168](#)
  - [\\_M\\_invalidate\\_all, 1168](#)
  - [\\_M\\_iterators, 1169](#)
  - [\\_M\\_local\\_iterators, 1169](#)
  - [\\_M\\_revalidate\\_singular, 1168](#)
  - [\\_M\\_swap, 1168, 1169](#)
  - [\\_M\\_version, 1170](#)
  - [~\\_Safe\\_unordered\\_container\\_base, 1167](#)
- [\\_\\_gnu\\_debug:: Safe\\_vector< \\_SafeSequence, \\_BaseSequence >, 1170](#)
- [\\_\\_gnu\\_debug:: Sequence\\_traits< \\_Sequence >, 1171](#)
- [\\_\\_gnu\\_debug:: basic\\_string< \\_CharT, \\_Traits, \\_Allocator >, 1171](#)
  - [\\_M\\_const\\_iterators, 1200](#)
  - [\\_M\\_detach\\_all, 1176](#)
  - [\\_M\\_detach\\_singular, 1176](#)
  - [\\_M\\_get\\_mutex, 1176](#)
  - [\\_M\\_invalidate\\_all, 1176](#)
  - [\\_M\\_invalidate\\_if, 1177](#)
  - [\\_M\\_iterators, 1200](#)
  - [\\_M\\_revalidate\\_singular, 1177](#)
  - [\\_M\\_swap, 1177](#)
  - [\\_M\\_transfer\\_from\\_if, 1177](#)
  - [\\_M\\_version, 1200](#)
- [append, 1177, 1178](#)
- [assign, 1179, 1180](#)
- [at, 1180, 1181](#)
- [back, 1181, 1182](#)
- [capacity, 1182](#)
- [compare, 1182, 1183](#)
- [empty, 1184](#)
- [erase, 1184](#)
- [find, 1185](#)
- [find\\_first\\_not\\_of, 1185](#)
- [find\\_first\\_of, 1186](#)
- [find\\_last\\_not\\_of, 1186](#)
- [find\\_last\\_of, 1187](#)
- [front, 1187, 1188](#)
- [get\\_allocator, 1188](#)
- [insert, 1188–1191](#)
- [length, 1192](#)
- [max\\_size, 1192](#)
- [npos, 1200](#)
- [operator+=, 1192](#)
- [replace, 1193–1197](#)
- [reserve, 1198](#)
- [rfind, 1199](#)
- [size, 1199](#)
- [swap, 1199](#)
- [\\_\\_gnu\\_internal, 494](#)
- [\\_\\_gnu\\_parallel, 495](#)
  - [\\_AlgorithmStrategy, 503](#)
  - [\\_BinIndex, 503](#)
  - [\\_CASable, 503](#)
  - [\\_CASable\\_bits, 554](#)
  - [\\_CASable\\_mask, 555](#)
  - [\\_FindAlgorithm, 504](#)
  - [\\_MultiwayMergeAlgorithm, 504](#)
  - [\\_Parallelism, 504](#)
  - [\\_PartialSumAlgorithm, 504](#)
  - [\\_SequenceIndex, 503](#)
  - [\\_SortAlgorithm, 505](#)
  - [\\_SplittingAlgorithm, 505](#)
  - [\\_ThreadIndex, 503](#)
  - [\\_\\_calc\\_borders, 505](#)
  - [\\_\\_compare\\_and\\_swap, 506](#)
  - [\\_\\_decode2, 506](#)
  - [\\_\\_determine\\_samples, 507](#)
  - [\\_\\_encode2, 507](#)
  - [\\_\\_equally\\_split, 508](#)
  - [\\_\\_equally\\_split\\_point, 508](#)
  - [\\_\\_fetch\\_and\\_add, 509](#)
  - [\\_\\_find\\_template, 509–511](#)
  - [\\_\\_for\\_each\\_template\\_random\\_access, 512](#)
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_ed, 514](#)
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop, 515](#)
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\\_static, 515](#)
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing, 516](#)
  - [\\_\\_is\\_sorted, 517](#)
  - [\\_\\_median\\_of\\_three\\_iterators, 517](#)
  - [\\_\\_merge\\_advance, 518](#)
  - [\\_\\_merge\\_advance\\_movc, 519](#)
  - [\\_\\_merge\\_advance\\_usual, 519](#)
  - [\\_\\_parallel\\_merge\\_advance, 520, 521](#)
  - [\\_\\_parallel\\_nth\\_element, 522](#)
  - [\\_\\_parallel\\_partial\\_sort, 522](#)
  - [\\_\\_parallel\\_partial\\_sum, 523](#)
  - [\\_\\_parallel\\_partial\\_sum\\_basecase, 523](#)
  - [\\_\\_parallel\\_partial\\_sum\\_linear, 524](#)
  - [\\_\\_parallel\\_partition, 524](#)
  - [\\_\\_parallel\\_random\\_shuffle, 525](#)
  - [\\_\\_parallel\\_random\\_shuffle\\_drs, 526](#)
  - [\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu, 526](#)
  - [\\_\\_parallel\\_sort, 527–531](#)
  - [\\_\\_parallel\\_sort\\_qs, 532](#)
  - [\\_\\_parallel\\_sort\\_qs\\_conquer, 533](#)
  - [\\_\\_parallel\\_sort\\_qs\\_divide, 533](#)
  - [\\_\\_parallel\\_sort\\_qsb, 534](#)

- `__parallel_unique_copy`, [534](#), [535](#)
- `__qsb_conquer`, [535](#)
- `__qsb_divide`, [536](#)
- `__qsb_local_sort_with_helping`, [537](#)
- `__random_number_pow2`, [537](#)
- `__rd_log2`, [538](#)
- `__round_up_to_pow2`, [538](#)
- `__search_template`, [538](#)
- `__sequential_multiway_merge`, [539](#)
- `__sequential_random_shuffle`, [540](#)
- `__shrink`, [540](#)
- `__shrink_and_double`, [541](#)
- `__yield`, [541](#)
- `list_partition`, [541](#)
- `max`, [542](#)
- `min`, [542](#)
- `multiseq_partition`, [543](#)
- `multiseq_selection`, [543](#)
- `multiway_merge`, [544](#)
- `multiway_merge_3_variant`, [546](#)
- `multiway_merge_4_variant`, [547](#)
- `multiway_merge_exact_splitting`, [547](#)
- `multiway_merge_loser_tree`, [548](#)
- `multiway_merge_loser_tree_sentinel`, [549](#)
- `multiway_merge_loser_tree_unguarded`, [549](#)
- `multiway_merge_sampling_splitting`, [550](#)
- `multiway_merge_sentinels`, [551](#)
- `parallel_balanced`, [504](#)
- `parallel_multiway_merge`, [553](#)
- `parallel_omp_loop`, [504](#)
- `parallel_omp_loop_static`, [504](#)
- `parallel_sort_mwms`, [553](#)
- `parallel_sort_mwms_pu`, [554](#)
- `parallel_taskqueue`, [504](#)
- `parallel_unbalanced`, [504](#)
- `sequential`, [504](#)
- `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>`, [1247](#)
- `_M_bins_begin`, [1247](#)
- `_M_num_threads`, [1248](#)
- `_M_sd`, [1248](#)
- `_M_seed`, [1248](#)
- `bins_end`, [1247](#)
- `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>`, [1244](#)
- `DRandomShufflingGlobalData`, [1245](#)
- `_M_bin_proc`, [1245](#)
- `_M_dist`, [1245](#)
- `_M_num_bins`, [1245](#)
- `_M_num_bits`, [1246](#)
- `_M_source`, [1246](#)
- `_M_starts`, [1246](#)
- `_M_temporaries`, [1246](#)
- `__gnu_parallel::DummyReduct`, [1249](#)
- `__gnu_parallel::EqualFromLess<_T1, _T2, _Compare>`, [1249](#)
- `first_argument_type`, [1250](#)
- `result_type`, [1250](#)
- `second_argument_type`, [1250](#)
- `__gnu_parallel::EqualTo<_T1, _T2>`, [1251](#)
- `first_argument_type`, [1251](#)
- `result_type`, [1252](#)
- `second_argument_type`, [1252](#)
- `__gnu_parallel::GuardedIterator<_RAIter, _Compare>`, [1252](#)
- `_GuardedIterator`, [1253](#)
- `operator_RAIter`, [1253](#)
- `operator<`, [1254](#)
- `operator<=`, [1255](#)
- `operator*`, [1254](#)
- `operator++`, [1254](#)
- `__gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>`, [1256](#)
- `_PCCFP`, [1257](#)
- `_PCCP`, [1257](#)
- `first`, [1258](#)
- `second`, [1258](#)
- `second_type`, [1257](#)
- `__gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>`, [1258](#)
- `__gnu_parallel::Job<_DifferenceTp>`, [1259](#)
- `_M_first`, [1259](#)
- `_M_last`, [1260](#)
- `_M_load`, [1260](#)
- `__gnu_parallel::Less<_T1, _T2>`, [1261](#)
- `first_argument_type`, [1261](#)
- `result_type`, [1262](#)
- `second_argument_type`, [1262](#)
- `__gnu_parallel::Lexicographic<_T1, _T2, _Compare>`, [1262](#)
- `first_argument_type`, [1263](#)
- `result_type`, [1263](#)
- `second_argument_type`, [1263](#)
- `__gnu_parallel::LexicographicReverse<_T1, _T2, _Compare>`, [1264](#)
- `first_argument_type`, [1265](#)
- `result_type`, [1265](#)
- `second_argument_type`, [1265](#)
- `__gnu_parallel::LoserTree<__stable, _Tp, _Compare>`, [1266](#)
- `_M_log_k`, [1268](#)
- `delete_min_insert`, [1267](#)
- `get_min_source`, [1267](#)
- `insert_start`, [1267](#)
- `__gnu_parallel::LoserTree<false, _Tp, _Compare>`, [1268](#)
- `delete_min_insert`, [1269](#)
- `get_min_source`, [1269](#)

[\\_\\_init\\_winner, 1270](#)  
[\\_\\_insert\\_start, 1270](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >, 1271](#)  
[\\_LoserTreeBase, 1272](#)  
[\\_M\\_comp, 1274](#)  
[\\_M\\_first\\_insert, 1274](#)  
[\\_M\\_log\\_k, 1274](#)  
[\\_M\\_losers, 1274](#)  
[\\_\\_get\\_min\\_source, 1273](#)  
[\\_\\_insert\\_start, 1273](#)  
[~\\_LoserTreeBase, 1272](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreeBase< \\_Tp, \\_Compare >::\\_Loser, 1275](#)  
[\\_M\\_key, 1275](#)  
[\\_M\\_source, 1275](#)  
[\\_M\\_sup, 1276](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >, 1276](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointer< false, \\_Tp, \\_Compare >, 1277](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >, 1278](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_Loser, 1279](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >, 1280](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >, 1281](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >, 1282](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreeTraits< \\_Tp >, 1283](#)  
[\\_M\\_use\\_pointer, 1284](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >, 1284](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >, 1285](#)  
[\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >, 1286](#)  
[\\_\\_gnu\\_parallel::\\_Multiplies< \\_Tp1, \\_Tp2, \\_Result >, 1287](#)  
[first\\_argument\\_type, 1288](#)  
[result\\_type, 1288](#)  
[second\\_argument\\_type, 1288](#)  
[\\_\\_gnu\\_parallel::\\_Nothing, 1289](#)  
[operator\(\), 1289](#)  
[\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >, 1292](#)  
[\\_M\\_num\\_threads, 1293](#)  
[\\_M\\_offsets, 1293](#)  
[\\_M\\_pieces, 1293](#)  
[\\_M\\_samples, 1294](#)  
[\\_M\\_source, 1294](#)  
[\\_M\\_starts, 1294](#)  
[\\_M\\_temporary, 1294](#)  
[\\_\\_gnu\\_parallel::\\_Piece< \\_DifferenceTp >, 1290](#)  
[\\_M\\_begin, 1290](#)  
[\\_M\\_end, 1290](#)  
[\\_\\_gnu\\_parallel::\\_Plus< \\_Tp1, \\_Tp2, \\_Result >, 1291](#)  
[first\\_argument\\_type, 1292](#)  
[result\\_type, 1292](#)  
[second\\_argument\\_type, 1292](#)  
[\\_\\_gnu\\_parallel::\\_PseudoSequence< \\_Tp, \\_DifferenceTp >, 1295](#)  
[\\_PseudoSequence, 1296](#)  
[begin, 1296](#)  
[end, 1296](#)  
[\\_\\_gnu\\_parallel::\\_PseudoSequenceIterator< \\_Tp, \\_DifferenceTp >, 1297](#)  
[\\_\\_gnu\\_parallel::\\_QSBThreadLocal< \\_RAIter >, 1297](#)  
[\\_M\\_elements\\_leftover, 1299](#)  
[\\_M\\_global, 1299](#)  
[\\_M\\_initial, 1299](#)  
[\\_M\\_leftover\\_parts, 1299](#)  
[\\_M\\_num\\_threads, 1300](#)  
[\\_Piece, 1298](#)  
[\\_QSBThreadLocal, 1298](#)  
[\\_\\_gnu\\_parallel::\\_RandomNumber, 1300](#)  
[\\_RandomNumber, 1300, 1301](#)  
[\\_\\_genrand\\_bits, 1301](#)  
[operator\(\), 1301, 1302](#)  
[\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< \\_Tp >, 1302](#)  
[\\_RestrictedBoundedConcurrentQueue, 1303](#)  
[~\\_RestrictedBoundedConcurrentQueue, 1303](#)  
[pop\\_back, 1303](#)  
[pop\\_front, 1303](#)  
[push\\_front, 1304](#)  
[\\_\\_gnu\\_parallel::\\_SamplingSorter< \\_\\_stable, \\_RAIter, \\_StrictWeakOrdering >, 1304](#)  
[\\_\\_gnu\\_parallel::\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >, 1305](#)  
[\\_\\_gnu\\_parallel::\\_Settings, 1305](#)  
[accumulate\\_minimal\\_n, 1307](#)  
[adjacent\\_difference\\_minimal\\_n, 1307](#)  
[cache\\_line\\_size, 1307](#)  
[count\\_minimal\\_n, 1308](#)  
[fill\\_minimal\\_n, 1308](#)  
[find\\_increasing\\_factor, 1308](#)  
[find\\_initial\\_block\\_size, 1308](#)  
[find\\_maximum\\_block\\_size, 1308](#)  
[find\\_scale\\_factor, 1309](#)  
[find\\_sequential\\_search\\_size, 1309](#)  
[for\\_each\\_minimal\\_n, 1309](#)  
[generate\\_minimal\\_n, 1309](#)  
[get, 1306](#)  
[L1\\_cache\\_size, 1309](#)  
[L2\\_cache\\_size, 1310](#)  
[max\\_element\\_minimal\\_n, 1310](#)  
[merge\\_minimal\\_n, 1310](#)

merge\_oversampling, 1310  
 min\_element\_minimal\_n, 1310  
 multiway\_merge\_minimal\_k, 1311  
 multiway\_merge\_minimal\_n, 1311  
 multiway\_merge\_oversampling, 1311  
 nth\_element\_minimal\_n, 1311  
 partial\_sort\_minimal\_n, 1311  
 partial\_sum\_dilation, 1312  
 partial\_sum\_minimal\_n, 1312  
 partition\_chunk\_share, 1312  
 partition\_chunk\_size, 1312  
 partition\_minimal\_n, 1312  
 qsb\_steals, 1313  
 random\_shuffle\_minimal\_n, 1313  
 replace\_minimal\_n, 1313  
 search\_minimal\_n, 1313  
 set, 1307  
 set\_difference\_minimal\_n, 1313  
 set\_intersection\_minimal\_n, 1314  
 set\_symmetric\_difference\_minimal\_n, 1314  
 set\_union\_minimal\_n, 1314  
 sort\_minimal\_n, 1314  
 sort\_mwms\_oversampling, 1314  
 sort\_qs\_num\_samples\_preset, 1315  
 sort\_qsb\_base\_case\_maximal\_n, 1315  
 TLB\_size, 1315  
 transform\_minimal\_n, 1315  
 unique\_copy\_minimal\_n, 1316  
 \_\_gnu\_parallel:: SplitConsistently< \_\_exact, \_RAIter, \_Compare, \_SortingPlacesIterator >, 1316  
 \_\_gnu\_parallel:: SplitConsistently< false, \_RAIter, \_Compare, \_SortingPlacesIterator >, 1316  
 \_\_gnu\_parallel:: SplitConsistently< true, \_RAIter, \_Compare, \_SortingPlacesIterator >, 1317  
 \_\_gnu\_parallel:: accumulate\_binop\_reduct< \_BinOp >, 1201  
 \_\_gnu\_parallel:: accumulate\_selector< \_It >, 1202  
     \_M\_finish\_iterator, 1203  
     operator(), 1202  
 \_\_gnu\_parallel:: adjacent\_difference\_selector< \_It >, 1203  
     \_M\_finish\_iterator, 1204  
 \_\_gnu\_parallel:: adjacent\_find\_selector, 1204  
     \_M\_sequential\_algorithm, 1205  
     operator(), 1205  
 \_\_gnu\_parallel:: binder1st< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 1206  
     argument\_type, 1207  
     result\_type, 1207  
 \_\_gnu\_parallel:: binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 1208  
     argument\_type, 1209  
     result\_type, 1209  
 \_\_gnu\_parallel:: \_\_count\_if\_selector< \_It, \_Diff >, 1209  
     \_M\_finish\_iterator, 1211  
     operator(), 1210  
 \_\_gnu\_parallel:: \_\_count\_selector< \_It, \_Diff >, 1211  
     \_M\_finish\_iterator, 1212  
     operator(), 1212  
 \_\_gnu\_parallel:: \_\_fill\_selector< \_It >, 1213  
     \_M\_finish\_iterator, 1214  
     operator(), 1213  
 \_\_gnu\_parallel:: \_\_find\_first\_of\_selector< \_FIterator >, 1214  
     \_M\_sequential\_algorithm, 1215  
     operator(), 1216  
 \_\_gnu\_parallel:: \_\_find\_if\_selector, 1216  
     \_M\_sequential\_algorithm, 1217  
     operator(), 1217  
 \_\_gnu\_parallel:: \_\_for\_each\_selector< \_It >, 1218  
     \_M\_finish\_iterator, 1219  
     operator(), 1219  
 \_\_gnu\_parallel:: \_\_generate\_selector< \_It >, 1220  
     \_M\_finish\_iterator, 1221  
     operator(), 1220  
 \_\_gnu\_parallel:: \_\_generic\_find\_selector, 1221  
 \_\_gnu\_parallel:: \_\_generic\_for\_each\_selector< \_It >, 1223  
     \_M\_finish\_iterator, 1224  
 \_\_gnu\_parallel:: \_\_identity\_selector< \_It >, 1224  
     \_M\_finish\_iterator, 1226  
     operator(), 1225  
 \_\_gnu\_parallel:: \_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 1226  
     \_M\_finish\_iterator, 1228  
     \_\_begin1\_iterator, 1228  
     \_\_begin2\_iterator, 1228  
     \_\_inner\_product\_selector, 1227  
     operator(), 1227  
 \_\_gnu\_parallel:: \_\_max\_element\_reduct< \_Compare, \_It >, 1229  
 \_\_gnu\_parallel:: \_\_min\_element\_reduct< \_Compare, \_It >, 1229  
 \_\_gnu\_parallel:: \_\_mismatch\_selector, 1230  
     \_M\_sequential\_algorithm, 1230  
     operator(), 1231  
 \_\_gnu\_parallel:: \_\_multiway\_merge\_3\_variant\_sentinel\_switch< \_\_sentinels, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 1231  
 \_\_gnu\_parallel:: \_\_multiway\_merge\_3\_variant\_sentinel\_switch< true, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 1232  
 \_\_gnu\_parallel:: \_\_multiway\_merge\_4\_variant\_sentinel\_switch< \_\_sentinels, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare >, 1232  
 \_\_gnu\_parallel:: \_\_multiway\_merge\_4\_variant\_sentinel\_switch<



[true, \\_RAIterlterator, \\_RAIter3, \\_DifferenceTp, \\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag, 1330](#)  
[\\_\\_Compare >, 1233](#)  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_\\_stable, \\_RAIterlterator, \\_RAIter3, \\_DifferenceTp, \\_\\_Compare >, 1233](#)  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAIterlterator, \\_RAIter3, \\_DifferenceTp, \\_\\_Compare >, 1234](#)  
[\\_\\_gnu\\_parallel::\\_\\_replace\\_if\\_selector< \\_It, \\_Op, \\_Tp >, 1235](#)  
[\\_\\_M\\_finish\\_iterator, 1236](#)  
[\\_\\_new\\_val, 1236](#)  
[\\_\\_replace\\_if\\_selector, 1235](#)  
[operator\(\), 1236](#)  
[\\_\\_gnu\\_parallel::\\_\\_replace\\_selector< \\_It, \\_Tp >, 1237](#)  
[\\_\\_M\\_finish\\_iterator, 1239](#)  
[\\_\\_new\\_val, 1238](#)  
[\\_\\_replace\\_selector, 1238](#)  
[operator\(\), 1238](#)  
[\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector< \\_It >, 1239](#)  
[\\_\\_M\\_finish\\_iterator, 1240](#)  
[operator\(\), 1240](#)  
[\\_\\_gnu\\_parallel::\\_\\_transform2\\_selector< \\_It >, 1241](#)  
[\\_\\_M\\_finish\\_iterator, 1242](#)  
[operator\(\), 1242](#)  
[\\_\\_gnu\\_parallel::\\_\\_unary\\_negate< \\_Predicate, argument\\_type >, 1243](#)  
[argument\\_type, 1243](#)  
[result\\_type, 1244](#)  
[\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag, 1317](#)  
[\\_\\_get\\_num\\_threads, 1318](#)  
[set\\_num\\_threads, 1318](#)  
[\\_\\_gnu\\_parallel::balanced\\_tag, 1319](#)  
[\\_\\_get\\_num\\_threads, 1319](#)  
[set\\_num\\_threads, 1319](#)  
[\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag, 1320](#)  
[\\_\\_gnu\\_parallel::default\\_parallel\\_tag, 1321](#)  
[\\_\\_get\\_num\\_threads, 1321](#)  
[set\\_num\\_threads, 1321](#)  
[\\_\\_gnu\\_parallel::equal\\_split\\_tag, 1322](#)  
[\\_\\_gnu\\_parallel::exact\\_tag, 1323](#)  
[\\_\\_get\\_num\\_threads, 1323](#)  
[set\\_num\\_threads, 1323](#)  
[\\_\\_gnu\\_parallel::find\\_tag, 1324](#)  
[\\_\\_gnu\\_parallel::growing\\_blocks\\_tag, 1325](#)  
[\\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag, 1325](#)  
[\\_\\_get\\_num\\_threads, 1326](#)  
[set\\_num\\_threads, 1326](#)  
[\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag, 1327](#)  
[\\_\\_get\\_num\\_threads, 1327](#)  
[set\\_num\\_threads, 1327](#)  
[\\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag, 1328](#)  
[\\_\\_get\\_num\\_threads, 1329](#)  
[set\\_num\\_threads, 1329](#)  
[\\_\\_gnu\\_parallel::omp\\_loop\\_tag, 1331](#)  
[\\_\\_get\\_num\\_threads, 1331](#)  
[\\_\\_gnu\\_parallel::parallel\\_tag, 1333](#)  
[\\_\\_get\\_num\\_threads, 1334](#)  
[parallel\\_tag, 1334](#)  
[set\\_num\\_threads, 1334](#)  
[\\_\\_gnu\\_parallel::quicksort\\_tag, 1335](#)  
[\\_\\_get\\_num\\_threads, 1336](#)  
[set\\_num\\_threads, 1336](#)  
[\\_\\_gnu\\_parallel::sampling\\_tag, 1337](#)  
[\\_\\_get\\_num\\_threads, 1337](#)  
[set\\_num\\_threads, 1337](#)  
[\\_\\_gnu\\_parallel::sequential\\_tag, 1338](#)  
[\\_\\_gnu\\_parallel::unbalanced\\_tag, 1338](#)  
[\\_\\_get\\_num\\_threads, 1339](#)  
[set\\_num\\_threads, 1339](#)  
[\\_\\_gnu\\_pbds, 555](#)  
[\\_\\_gnu\\_pbds::associative\\_tag, 1340](#)  
[\\_\\_gnu\\_pbds::basic\\_branch< Key, Mapped, Tag, Node\\_Update, Policy\\_Tl, \\_Alloc >, 1340](#)  
[\\_\\_gnu\\_pbds::basic\\_branch\\_tag, 1341](#)  
[\\_\\_gnu\\_pbds::basic\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Resize\\_Policy, Store\\_Hash, Tag, Policy\\_Tl, \\_Alloc >, 1342](#)  
[\\_\\_gnu\\_pbds::basic\\_hash\\_tag, 1343](#)  
[\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee, 1344](#)  
[\\_\\_gnu\\_pbds::binary\\_heap\\_tag, 1345](#)  
[\\_\\_gnu\\_pbds::binomial\\_heap\\_tag, 1346](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >, 1346](#)  
[cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger, 1348](#)  
[external\\_load\\_access, 1347](#)  
[get\\_load, 1348](#)  
[is\\_grow\\_needed, 1348](#)  
[is\\_resize\\_needed, 1348](#)  
[notify\\_cleared, 1349](#)  
[notify\\_erase\\_search\\_collision, 1349](#)  
[notify\\_erase\\_search\\_end, 1349](#)  
[notify\\_erase\\_search\\_start, 1349](#)  
[notify\\_erased, 1350](#)  
[notify\\_externally\\_resized, 1350](#)  
[notify\\_find\\_search\\_collision, 1350](#)  
[notify\\_find\\_search\\_end, 1350](#)  
[notify\\_find\\_search\\_start, 1351](#)  
[notify\\_insert\\_search\\_collision, 1351](#)  
[notify\\_insert\\_search\\_end, 1351](#)  
[notify\\_insert\\_search\\_start, 1351](#)  
[notify\\_inserted, 1352](#)  
[notify\\_resized, 1352](#)  
[set\\_load, 1352](#)



[\\_\\_gnu\\_pbds::cc\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Hash\\_Fn, Resize\\_Policy, Store\\_Hash, \\_Alloc >, 1353](#)  
[cc\\_hash\\_table, 1354–1357](#)  
[\\_\\_gnu\\_pbds::cc\\_hash\\_tag, 1358](#)  
[\\_\\_gnu\\_pbds::container\\_error, 1359](#)  
[what, 1360](#)  
[\\_\\_gnu\\_pbds::container\\_tag, 1360](#)  
[\\_\\_gnu\\_pbds::container\\_traits< Cntnr >, 1361](#)  
[erase\\_can\\_throw, 1362](#)  
[order\\_preserving, 1362](#)  
[reverse\\_iteration, 1362](#)  
[split\\_join\\_can\\_throw, 1362](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< \\_Tag >, 1362](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< binary\\_heap\\_tag >, 1362](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< binomial\\_heap\\_tag >, 1363](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< cc\\_hash\\_tag >, 1363](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< gp\\_hash\\_tag >, 1364](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >, 1364](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >, 1365](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >, 1365](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >, 1366](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >, 1366](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >, 1367](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >, 1367](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >, 1368](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >, 1368](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it< Node, Const\\_Iterator, Iterator, \\_Alloc >, 1370](#)  
[const\\_reference, 1371](#)  
[difference\\_type, 1371](#)  
[get\\_l\\_child, 1372](#)  
[get\\_metadata, 1372](#)  
[get\\_r\\_child, 1373](#)  
[iterator\\_category, 1371](#)  
[metadata\\_const\\_reference, 1371](#)  
[metadata\\_type, 1371](#)  
[operator!=, 1373](#)  
[operator\\*, 1373](#)  
[operator==, 1373](#)  
[reference, 1372](#)  
[value\\_type, 1372](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >, 1374](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it< Node, Const\\_Iterator, Iterator, \\_Alloc >, 1376](#)  
[const\\_reference, 1377](#)  
[difference\\_type, 1377](#)  
[get\\_l\\_child, 1378](#)  
[get\\_metadata, 1379](#)  
[get\\_r\\_child, 1379](#)  
[iterator\\_category, 1377](#)  
[metadata\\_const\\_reference, 1377](#)  
[metadata\\_type, 1378](#)  
[operator!=, 1379](#)  
[operator\\*, 1379](#)  
[operator==, 1380](#)  
[reference, 1378](#)  
[value\\_type, 1378](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >, 1380](#)  
[node\\_const\\_iterator, 1381](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >, 1381](#)  
[node\\_const\\_iterator, 1382](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >, 1382](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator< Value\\_Type, Entry, Simple, \\_Alloc >, 1384](#)  
[binary\\_heap\\_const\\_iterator\\_, 1387](#)  
[const\\_pointer, 1385](#)  
[const\\_reference, 1386](#)  
[difference\\_type, 1386](#)  
[iterator\\_category, 1386](#)  
[operator!=, 1388](#)  
[operator\\*, 1388](#)  
[operator->, 1388](#)  
[operator==, 1389](#)  
[pointer, 1386](#)  
[reference, 1387](#)  
[value\\_type, 1387](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator< Value\\_Type, Entry, Simple, \\_Alloc >, 1390](#)  
[binary\\_heap\\_point\\_const\\_iterator\\_, 1392, 1393](#)  
[const\\_pointer, 1391](#)  
[const\\_reference, 1391](#)  
[difference\\_type, 1391](#)  
[iterator\\_category, 1391](#)  
[operator!=, 1393](#)  
[operator\\*, 1393](#)  
[operator->, 1393](#)  
[operator==, 1394](#)

- pointer, [1392](#)
- reference, [1392](#)
- value\_type, [1392](#)
- \_\_gnu\_pbds::detail::binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc >, [1394](#)
- \_\_gnu\_pbds::detail::binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc >, [1396](#)
- \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Cltr, \_Alloc >, [1400](#)
- \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >, [1399](#)
- \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1401](#)
- empty, [1403](#)
- get\_comb\_hash\_fn, [1404](#)
- get\_eq\_fn, [1404](#)
- get\_hash\_fn, [1405](#)
- get\_resize\_policy, [1405](#)
- \_\_gnu\_pbds::detail::cond\_dealtor< Entry, \_Alloc >, [1406](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_TI >, [1407](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1411](#)
- type, [1411](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1411](#)
- type, [1412](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI >, [1412](#)
- type, [1413](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI >, [1413](#)
- type, [1413](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, pat\_trie\_tag, Policy\_TI >, [1414](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI >, [1414](#)
- type, [1414](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI >, [1415](#)
- type, [1415](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1416](#)
- type, [1416](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1416](#)
- type, [1417](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI >, [1417](#)
- type, [1418](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_TI >, [1418](#)
- type, [1418](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_TI >, [1419](#)
- type, [1419](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, rb\_tree\_tag, Policy\_TI >, [1419](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_TI >, [1420](#)
- type, [1420](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, [1407](#)
- type, [1407](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, [1408](#)
- type, [1408](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, [1408](#)
- type, [1409](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, [1409](#)
- type, [1410](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, [1410](#)
- type, [1410](#)
- \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [1421](#)
- type, [1421](#)
- \_\_gnu\_pbds::detail::default\_eq\_fn< Key >, [1421](#)
- type, [1422](#)
- \_\_gnu\_pbds::detail::default\_hash\_fn< Key >, [1422](#)
- type, [1422](#)
- \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn >, [1423](#)
- type, [1423](#)
- \_\_gnu\_pbds::detail::default\_resize\_policy< Comb\_Hash\_Fn >, [1423](#)
- type, [1424](#)
- \_\_gnu\_pbds::detail::default\_trie\_access\_traits< Key >, [1424](#)
- \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string<

- Char, Char\_Traits, std::allocator< char > > >, 1424
- type, 1425
- \_\_gnu\_pbds::detail::default\_update\_policy, 1425
- type, 1425
- \_\_gnu\_pbds::detail::dumnode\_const\_iterator< Key, Data, \_Alloc >, 1426
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, No\_Throw >, 1426
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >, 1426
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >::type, 1427
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, 1427
- type, 1428
- \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw >, 1428
- \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false >, 1428
- \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, true >, 1429
- \_\_gnu\_pbds::detail::eq\_by\_less< Key, Cmp\_Fn >, 1429
- \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, 1430
- empty, 1433
- get\_comb\_probe\_fn, 1433
- get\_eq\_fn, 1434
- get\_hash\_fn, 1434
- get\_probe\_fn, 1435
- get\_resize\_policy, 1435
- \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash >, 1436
- \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, false >, 1437
- \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, true >, 1437
- \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base<node\_begin, 1459, 1460
- Size\_Type, Hold\_Size >, 1438
- \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base<node\_end, 1460
- Size\_Type, true >, 1438
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, 1439
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, 1439
- Node, \_Alloc >, 1441
- const\_pointer, 1442
- const\_reference, 1442
- difference\_type, 1442
- iterator\_category, 1442
- left\_child\_next\_sibling\_heap\_const\_iterator\_, 1443, 1444
- operator!=, 1444
- operator\*, 1444
- operator->, 1445
- operator==, 1445
- pointer, 1443
- reference, 1443
- value\_type, 1443
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, 1446
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, 1447
- Node, \_Alloc >, 1447
- const\_pointer, 1448
- const\_reference, 1448
- difference\_type, 1448
- iterator\_category, 1448
- left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, 1449, 1450
- operator!=, 1450
- operator\*, 1450
- operator->, 1450
- operator==, 1451
- pointer, 1449
- reference, 1449
- value\_type, 1449
- \_\_gnu\_pbds::detail::lu\_counter\_metadata< Size\_Type >, 1451
- \_\_gnu\_pbds::detail::lu\_counter\_policy\_base< Size\_Type >, 1452
- \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >, 1452
- \_\_gnu\_pbds::detail::mask\_based\_range\_hashing< Size\_Type >, 1454
- \_\_gnu\_pbds::detail::mod\_based\_range\_hashing< Size\_Type >, 1455
- \_\_gnu\_pbds::detail::no\_throw\_copies< Key, Mapped >, 1456
- \_\_gnu\_pbds::detail::no\_throw\_copies< Key, null\_type >, 1457
- \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, 1457
- node\_end, 1460
- \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::cond\_dtor< Size\_Type >, 1461
- \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it< Value\_Type, Metadata\_Type, \_Alloc >, 1461
- get\_l\_child, 1463
- get\_r\_child, 1463
- \_\_gnu\_pbds::detail::ov\_tree\_node\_it< Value\_Type, Metadata\_Type, \_Alloc >, 1463
- get\_l\_child, 1464
- get\_r\_child, 1465
- operator\*, 1465
- \_\_gnu\_pbds::detail::pairing\_heap< Value\_Type, Cmp\_Fn, \_Alloc >, 1465

[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#), [1468](#)  
[node\\_type](#), [1468](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#), [1469](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head< \\_ATraits, Metadata >](#), [1471](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >](#), [1472](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::const\\_iterator](#), [1474](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::iterator](#), [1475](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_lter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#), [1476](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf< \\_ATraits, Metadata >](#), [1478](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Metadata, \\_Alloc >](#), [1479](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< null\\_type, \\_Alloc >](#), [1480](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base< \\_ATraits, Metadata >](#), [1481](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#), [1482](#)  
[\\_\\_rebind\\_m](#), [1483](#)  
[get\\_child](#), [1484](#)  
[get\\_metadata](#), [1484](#)  
[metadata\\_type](#), [1484](#)  
[num\\_children](#), [1484](#)  
[operator!=](#), [1485](#)  
[operator\\*](#), [1485](#)  
[operator==](#), [1485](#)  
[valid\\_prefix](#), [1485](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#), [1486](#)  
[\\_\\_rebind\\_m](#), [1487](#)  
[get\\_child](#), [1488](#)  
[get\\_metadata](#), [1488](#)  
[metadata\\_type](#), [1487](#)  
[num\\_children](#), [1488](#)  
[operator!=](#), [1488](#)  
[operator\\*](#), [1489](#)  
[operator==](#), [1489](#)  
[valid\\_prefix](#), [1489](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#), [1490](#)  
[node\\_begin](#), [1492](#)  
[node\\_end](#), [1493](#)  
[node\\_type](#), [1492](#)  
[\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base< \\_Alloc >](#), [1493](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, Store\\_Hash >](#), [1494](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >](#), [1494](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, true >](#), [1495](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, false >](#), [1496](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, true >](#), [1497](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, Store\\_Hash >](#), [1497](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >](#), [1498](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >](#), [1499](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >](#), [1499](#)  
[\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#), [1500](#)  
[node\\_begin](#), [1503](#)  
[node\\_end](#), [1504](#)  
[\\_\\_gnu\\_pbds::detail::rb\\_tree\\_node< Value\\_Type, Metadata, \\_Alloc >](#), [1504](#)  
[\\_\\_gnu\\_pbds::detail::rc< \\_Node, \\_Alloc >](#), [1505](#)  
[\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#), [1506](#)  
[\\_\\_gnu\\_pbds::detail::resize\\_policy< \\_Tp >](#), [1508](#)  
[\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#), [1509](#)  
[node\\_begin](#), [1512](#)  
[node\\_end](#), [1512](#), [1513](#)  
[\\_\\_gnu\\_pbds::detail::splay\\_tree\\_node< Value\\_Type, Metadata, \\_Alloc >](#), [1513](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th >](#), [1514](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, null\\_type >](#), [1515](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_hash< \\_Th >](#), [1516](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_value< \\_Tv >](#), [1517](#)  
[\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >](#), [1518](#)  
[\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#), [1518](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, \\_BTp >](#), [1521](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, false >](#), [1521](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, true >](#), [1521](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#), [1522](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Data, Cmp\\_Fn, Node\\_Update, Tag, \\_Alloc >](#), [1522](#)

- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [1523](#)
  - node\_const\_iterator, [1523](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [1524](#)
  - node\_const\_iterator, [1525](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [1525](#)
  - node\_const\_iterator, [1526](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [1527](#)
  - node\_const\_iterator, [1527](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [1528](#)
  - node\_const\_iterator, [1529](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [1529](#)
  - node\_const\_iterator, [1530](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, BTp >, [1531](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, false >, [1531](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, true >, [1531](#)
- \_\_gnu\_pbds::detail::trie\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc >, [1532](#)
- \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, [1532](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, Data, ATraits, Node\_Update, Tag, \_Alloc >, [1534](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1534](#)
  - node\_const\_iterator, [1535](#)
  - node\_update, [1535](#)
  - synth\_access\_traits, [1535](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1536](#)
  - node\_const\_iterator, [1536](#)
  - node\_update, [1536](#)
  - synth\_access\_traits, [1537](#)
- \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc, Store\_Hash >, [1537](#)
- \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc, false >, [1538](#)
- \_\_gnu\_pbds::detail::type\_base< Key, Mapped, \_Alloc, true >, [1539](#)
- \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc, false >, [1539](#)
- \_\_gnu\_pbds::detail::type\_base< Key, null\_type, \_Alloc, true >, [1540](#)
- \_\_gnu\_pbds::detail::type\_dispatch< Key, Mapped, \_Alloc, Store\_Hash >, [1541](#)
- \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc, Store\_Hash >, [1541](#)
- \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size\_Type >, [1542](#)
  - operator(), [1543](#)
- \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size\_Type >, [1543](#)
  - operator(), [1544](#)
- \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, [1545](#)
  - gp\_hash\_table, [1546–1550](#)
- \_\_gnu\_pbds::gp\_hash\_tag, [1552](#)
- \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type >, [1552](#)
  - hash\_exponential\_size\_policy, [1553](#)
- \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1553](#)
  - external\_load\_access, [1555](#)
  - get\_loads, [1555](#)
  - hash\_load\_check\_resize\_trigger, [1555](#)
  - notify\_cleared, [1555](#)
  - notify\_inserted, [1555](#)
  - notify\_resized, [1556](#)
  - set\_loads, [1556](#)
- \_\_gnu\_pbds::hash\_prime\_size\_policy, [1556](#)
  - hash\_prime\_size\_policy, [1557](#)
  - size\_type, [1557](#)
- \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [1557](#)
  - get\_actual\_size, [1559](#)
  - get\_new\_size, [1560](#)
  - get\_size\_policy, [1560](#)
  - get\_trigger\_policy, [1560](#), [1561](#)
  - hash\_standard\_resize\_policy, [1559](#)
  - resize, [1561](#)
- \_\_gnu\_pbds::insert\_error, [1562](#)
  - what, [1562](#)
- \_\_gnu\_pbds::join\_error, [1563](#)
  - what, [1564](#)
- \_\_gnu\_pbds::linear\_probe\_fn< Size\_Type >, [1564](#)
  - operator(), [1565](#)
- \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, [1565](#)
  - list\_update, [1566](#)
- \_\_gnu\_pbds::list\_update\_tag, [1567](#)
- \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, [1567](#)
  - max\_count, [1569](#)
  - metadata\_reference, [1568](#)
  - metadata\_type, [1568](#)
  - operator(), [1569](#)
- \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, [1570](#)
  - metadata\_reference, [1570](#)
  - metadata\_type, [1571](#)

- operator(), 1571
- \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_Tp4 >, 1572
- \_\_gnu\_pbds::null\_type, 1572
- \_\_gnu\_pbds::ov\_tree\_tag, 1573
- \_\_gnu\_pbds::pairing\_heap\_tag, 1574
- \_\_gnu\_pbds::pat\_trie\_tag, 1575
- \_\_gnu\_pbds::point\_invalidation\_guarantee, 1576
- \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc >, 1576
- \_\_gnu\_pbds::priority\_queue\_tag, 1578
- \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type >, 1578
  - operator(), 1579
- \_\_gnu\_pbds::range\_invalidation\_guarantee, 1579
- \_\_gnu\_pbds::rb\_tree\_tag, 1580
- \_\_gnu\_pbds::rc\_binomial\_heap\_tag, 1581
- \_\_gnu\_pbds::resize\_error, 1582
  - what, 1582
- \_\_gnu\_pbds::sample\_probe\_fn, 1583
  - operator(), 1584
  - sample\_probe\_fn, 1583
  - swap, 1584
- \_\_gnu\_pbds::sample\_range\_hashing, 1584
  - notify\_resized, 1585
  - operator(), 1586
  - sample\_range\_hashing, 1585
  - size\_type, 1585
  - swap, 1586
- \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 1586
  - notify\_resized, 1587
  - operator(), 1587
  - sample\_ranged\_hash\_fn, 1587
  - swap, 1587
- \_\_gnu\_pbds::sample\_ranged\_probe\_fn, 1588
- \_\_gnu\_pbds::sample\_resize\_policy, 1588
  - get\_new\_size, 1590
  - is\_resize\_needed, 1590
  - notify\_cleared, 1590
  - notify\_erase\_search\_collision, 1590
  - notify\_erase\_search\_end, 1590
  - notify\_erase\_search\_start, 1590
  - notify\_erased, 1591
  - notify\_find\_search\_collision, 1591
  - notify\_find\_search\_end, 1591
  - notify\_find\_search\_start, 1591
  - notify\_insert\_search\_collision, 1591
  - notify\_insert\_search\_end, 1591
  - notify\_insert\_search\_start, 1592
  - notify\_inserted, 1592
  - notify\_resized, 1592
  - sample\_range\_hashing, 1592
  - sample\_resize\_policy, 1589
  - size\_type, 1589
  - swap, 1592
- \_\_gnu\_pbds::sample\_resize\_trigger, 1593
  - is\_grow\_needed, 1594
  - is\_resize\_needed, 1594
  - notify\_cleared, 1594
  - notify\_erase\_search\_collision, 1594
  - notify\_erase\_search\_end, 1594
  - notify\_erase\_search\_start, 1595
  - notify\_erased, 1595
  - notify\_externally\_resized, 1595
  - notify\_find\_search\_collision, 1595
  - notify\_find\_search\_end, 1595
  - notify\_find\_search\_start, 1595
  - notify\_insert\_search\_collision, 1596
  - notify\_insert\_search\_end, 1596
  - notify\_insert\_search\_start, 1596
  - notify\_inserted, 1596
  - notify\_resized, 1596
  - sample\_range\_hashing, 1596
  - sample\_resize\_trigger, 1594
  - size\_type, 1593
  - swap, 1597
- \_\_gnu\_pbds::sample\_size\_policy, 1597
  - get\_nearest\_larger\_size, 1598
  - get\_nearest\_smaller\_size, 1598
  - sample\_range\_hashing, 1598
  - sample\_size\_policy, 1598
  - size\_type, 1597
  - swap, 1598
- \_\_gnu\_pbds::sample\_tree\_node\_update< Const\_Node\_Iter, Node\_Iter, Cmp\_Fn, \_Alloc >, 1599
- \_\_gnu\_pbds::sample\_trie\_access\_traits, 1599
  - begin, 1600
  - e\_pos, 1600
  - e\_type, 1600
  - end, 1600
- \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, 1601
  - operator(), 1601
  - sample\_trie\_node\_update, 1601
- \_\_gnu\_pbds::sample\_update\_policy, 1602
  - metadata\_type, 1602
  - operator(), 1603
  - sample\_update\_policy, 1602
  - swap, 1603
- \_\_gnu\_pbds::sequence\_tag, 1604
- \_\_gnu\_pbds::splay\_tree\_tag, 1605
- \_\_gnu\_pbds::string\_tag, 1606
- \_\_gnu\_pbds::thin\_heap\_tag, 1607
- \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >, 1607
  - cmp\_fn, 1608
  - tree, 1609
- \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc >, 1610



[find\\_by\\_order](#), [1611](#), [1612](#)  
[operator\(\)](#), [1612](#)  
[order\\_of\\_key](#), [1612](#)  
[\\_\\_gnu\\_pbds::tree\\_tag](#), [1613](#)  
[\\_\\_gnu\\_pbds::trie< Key, Mapped, \\_ATraits, Tag, Node\\_Update, \\_Alloc >](#), [1613](#)  
[access\\_traits](#), [1614](#)  
[trie](#), [1615](#)  
[\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, \\_ATraits, \\_Alloc >](#), [1616](#)  
[find\\_by\\_order](#), [1618](#)  
[operator\(\)](#), [1618](#)  
[order\\_of\\_key](#), [1618](#)  
[order\\_of\\_prefix](#), [1619](#)  
[\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update< Node\\_Cltr, Node\\_Itr, \\_ATraits, \\_Alloc >](#), [1620](#)  
[a\\_const\\_iterator](#), [1621](#)  
[access\\_traits](#), [1622](#)  
[allocator\\_type](#), [1622](#)  
[operator\(\)](#), [1622](#)  
[prefix\\_range](#), [1623](#)  
[size\\_type](#), [1622](#)  
[\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits< String, Min\\_E\\_Val, Max\\_E\\_Val, Reverse, \\_Alloc >](#), [1624](#)  
[begin](#), [1625](#)  
[const\\_iterator](#), [1625](#)  
[e\\_pos](#), [1626](#)  
[e\\_type](#), [1625](#)  
[end](#), [1626](#)  
[\\_\\_gnu\\_pbds::trie\\_tag](#), [1627](#)  
[\\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#), [1627](#)  
[\\_\\_gnu\\_profile](#), [557](#)  
[\\_GLIBCXX\\_PROFILE\\_DEFINE\\_UNINIT\\_DATA](#), [561](#)  
[\\_\\_env\\_t](#), [561](#)  
[\\_\\_profcxx\\_init](#), [561](#)  
[\\_\\_report](#), [561](#)  
[\\_\\_gnu\\_profile::\\_\\_container\\_size\\_info](#), [1628](#)  
[\\_\\_gnu\\_profile::\\_\\_container\\_size\\_stack\\_info](#), [1629](#)  
[\\_\\_gnu\\_profile::\\_\\_hashfunc\\_info](#), [1630](#)  
[\\_\\_gnu\\_profile::\\_\\_hashfunc\\_stack\\_info](#), [1631](#)  
[\\_\\_gnu\\_profile::\\_\\_list2vector\\_info](#), [1632](#)  
[\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#), [1633](#)  
[\\_\\_gnu\\_profile::\\_\\_map2umap\\_stack\\_info](#), [1634](#)  
[\\_\\_gnu\\_profile::\\_\\_object\\_info\\_base](#), [1635](#)  
[\\_\\_gnu\\_profile::\\_\\_reentrance\\_guard](#), [1636](#)  
[\\_\\_gnu\\_profile::\\_\\_stack\\_hash](#), [1637](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_base< \\_\\_object\\_info, \\_\\_stack\\_info >](#), [1637](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_container\\_size](#), [1638](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func](#), [1639](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size](#), [1640](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_map2umap](#), [1641](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size](#), [1642](#)  
[\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list](#), [1643](#)  
[\\_\\_gnu\\_profile::\\_\\_vector2list\\_info](#), [1644](#)  
[\\_\\_gnu\\_profile::\\_\\_vector2list\\_stack\\_info](#), [1645](#)  
[\\_\\_gnu\\_profile::\\_\\_warning\\_data](#), [1646](#)  
[\\_\\_gnu\\_sequential](#), [562](#)  
[\\_\\_heap\\_select](#)  
[std](#), [676](#)  
[\\_\\_init\\_winner](#)  
[\\_\\_gnu\\_parallel::LoserTree< false, \\_Tp, \\_Compare >](#), [1270](#)  
[\\_\\_inner\\_product\\_selector](#)  
[\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector< \\_It, \\_It2, \\_Tp >](#), [1227](#)  
[\\_\\_inplace\\_stable\\_sort](#)  
[std](#), [676](#)  
[\\_\\_insert\\_start](#)  
[\\_\\_gnu\\_parallel::LoserTree< \\_\\_stable, \\_Tp, \\_Compare >](#), [1267](#)  
[\\_\\_gnu\\_parallel::LoserTree< false, \\_Tp, \\_Compare >](#), [1270](#)  
[\\_\\_gnu\\_parallel::LoserTreeBase< \\_Tp, \\_Compare >](#), [1273](#)  
[\\_\\_insertion\\_sort](#)  
[std](#), [676](#)  
[\\_\\_introsort\\_loop](#)  
[std](#), [677](#)  
[\\_\\_invoke](#)  
[Utilities](#), [437](#)  
[\\_\\_ioint](#)  
[std](#), [788](#)  
[\\_\\_is\\_sorted](#)  
[\\_\\_gnu\\_parallel](#), [517](#)  
[\\_\\_iterator\\_category](#)  
[Iterators](#), [125](#)  
[\\_\\_lg](#)  
[std](#), [677](#)  
[\\_\\_match\\_flag](#)  
[std::regex\\_constants](#), [840](#)  
[\\_\\_median](#)  
[SGI](#), [374](#), [375](#)  
[\\_\\_median\\_of\\_three\\_iterators](#)  
[\\_\\_gnu\\_parallel](#), [517](#)  
[\\_\\_merge\\_adaptive](#)  
[std](#), [677](#)  
[\\_\\_merge\\_advance](#)  
[\\_\\_gnu\\_parallel](#), [518](#)  
[\\_\\_merge\\_advance\\_movc](#)  
[\\_\\_gnu\\_parallel](#), [519](#)  
[\\_\\_merge\\_advance\\_usual](#)  
[\\_\\_gnu\\_parallel](#), [519](#)  
[\\_\\_merge\\_without\\_buffer](#)  
[std](#), [678](#)  
[\\_\\_move\\_median\\_to\\_first](#)  
[std](#), [678](#)  
[\\_\\_move\\_merge](#)

- std, 678
- \_\_move\_merge\_adaptive
  - std, 679
- \_\_move\_merge\_adaptive\_backward
  - std, 679
- \_\_new\_val
  - \_\_gnu\_parallel::\_\_replace\_if\_selector< \_It, \_Op, \_Tp >, 1236
  - \_\_gnu\_parallel::\_\_replace\_selector< \_It, \_Tp >, 1238
- \_\_num\_bitmaps
  - \_\_gnu\_cxx::\_\_detail, 480
- \_\_num\_blocks
  - \_\_gnu\_cxx::\_\_detail, 481
- \_\_num\_get\_type
  - std::basic\_ios< \_CharT, \_Traits >, 2117
  - std::basic\_ofstream< \_CharT, \_Traits >, 2324
  - std::basic\_ostream< \_CharT, \_Traits >, 2368
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2412
- \_\_num\_put\_type
  - std::basic\_fstream< \_CharT, \_Traits >, 2000
  - std::basic\_ifstream< \_CharT, \_Traits >, 2065
  - std::basic\_ios< \_CharT, \_Traits >, 2118
  - std::basic\_iostream< \_CharT, \_Traits >, 2153
  - std::basic\_istream< \_CharT, \_Traits >, 2216
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2270
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2582
- \_\_once\_call
  - std, 788
- \_\_once\_callable
  - std, 789
- \_\_once\_proxy
  - std, 679
- \_\_parallel\_merge\_advance
  - \_\_gnu\_parallel, 520, 521
- \_\_parallel\_nth\_element
  - \_\_gnu\_parallel, 522
- \_\_parallel\_partial\_sort
  - \_\_gnu\_parallel, 522
- \_\_parallel\_partial\_sum
  - \_\_gnu\_parallel, 523
- \_\_parallel\_partial\_sum\_basecase
  - \_\_gnu\_parallel, 523
- \_\_parallel\_partial\_sum\_linear
  - \_\_gnu\_parallel, 524
- \_\_parallel\_partition
  - \_\_gnu\_parallel, 524
- \_\_parallel\_random\_shuffle
  - \_\_gnu\_parallel, 525
- \_\_parallel\_random\_shuffle\_drs
  - \_\_gnu\_parallel, 526
- \_\_parallel\_random\_shuffle\_drs\_pu
  - \_\_gnu\_parallel, 526
- \_\_parallel\_sort
  - \_\_gnu\_parallel, 527–531
- \_\_parallel\_sort\_qs
  - \_\_gnu\_parallel, 532
- \_\_parallel\_sort\_qs\_conquer
  - \_\_gnu\_parallel, 533
- \_\_parallel\_sort\_qs\_divide
  - \_\_gnu\_parallel, 533
- \_\_parallel\_sort\_qsb
  - \_\_gnu\_parallel, 534
- \_\_parallel\_unique\_copy
  - \_\_gnu\_parallel, 534, 535
- \_\_partition
  - std, 680
- \_\_polynomial
  - std::regex\_constants, 848
- \_\_profcxx\_init
  - \_\_gnu\_profile, 561
- \_\_ptr\_rebind
  - std, 669
- \_\_qsb\_conquer
  - \_\_gnu\_parallel, 535
- \_\_qsb\_divide
  - \_\_gnu\_parallel, 536
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, 537
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, 537
- \_\_rd\_log2
  - \_\_gnu\_parallel, 538
- \_\_rebind\_m
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 1483
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 1487
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_\_replace\_if\_selector< \_It, \_Op, \_Tp >, 1235
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_\_replace\_selector< \_It, \_Tp >, 1238
- \_\_report
  - \_\_gnu\_profile, 561
- \_\_reverse
  - std, 680, 681
- \_\_rotate
  - stl\_algo.h, 4347, 4348
- \_\_rotate\_adaptive
  - std, 681
- \_\_round\_up\_to\_pow2



- \_\_gnu\_parallel, 538
- \_\_sample
  - std, 681, 682
- \_\_search\_n\_aux
  - std, 682
- \_\_search\_template
  - \_\_gnu\_parallel, 538
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 539
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 540
- \_\_shared\_timed\_mutex\_base
  - Mutexes, 208
- \_\_shrink
  - \_\_gnu\_parallel, 540
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 541
- \_\_stable\_partition\_adaptive
  - std, 683
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 462
- \_\_streambuf\_type
  - std::basic\_streambuf< \_CharT, \_Traits >, 2467
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3937
- \_\_syntax\_option
  - std::regex\_constants, 841
- \_\_try\_to\_lock
  - std, 683
- \_\_umap\_traits
  - std, 669
- \_\_ummap\_traits
  - std, 669
- \_\_umset\_traits
  - std, 669
- \_\_unguarded\_insertion\_sort
  - std, 684
- \_\_unguarded\_linear\_insert
  - std, 684
- \_\_unguarded\_partition
  - std, 684
- \_\_unguarded\_partition\_pivot
  - std, 685
- \_\_unique\_copy
  - std, 685, 686
- \_\_uset\_traits
  - std, 669
- \_\_valid\_range
  - \_\_gnu\_debug, 492, 493
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 493, 494
- \_\_verbose\_terminate\_handler
  - Exceptions, 86
- \_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 894–897
- \_\_yield
  - \_\_gnu\_parallel, 541
- ~\_LoserTreeBase
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >, 1272
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, 1303
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::Safe\_sequence\_base, 1159
- ~\_Safe\_unordered\_container\_base
  - \_\_gnu\_debug::Safe\_unordered\_container\_base, 1167
- ~\_allocated\_ptr
  - std::\_\_allocated\_ptr< \_Alloc >, 1669
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 898
- ~any
  - std::experimental::fundamentals\_v1::any, 2912
- ~auto\_ptr
  - std::auto\_ptr< \_Tp >, 1950
- ~basic\_filebuf
  - std::basic\_filebuf< \_CharT, \_Traits >, 1968
- ~basic\_fstream
  - std::basic\_fstream< \_CharT, \_Traits >, 2004
- ~basic\_ifstream
  - std::basic\_ifstream< \_CharT, \_Traits >, 2070
- ~basic\_ios
  - std::basic\_ios< \_CharT, \_Traits >, 2122
- ~basic\_iostream
  - std::basic\_iostream< \_CharT, \_Traits >, 2156
- ~basic\_istream
  - std::basic\_istream< \_CharT, \_Traits >, 2219
- ~basic\_istreamstream
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2274
- ~basic\_ofstream
  - std::basic\_ofstream< \_CharT, \_Traits >, 2328
- ~basic\_ostream
  - std::basic\_ostream< \_CharT, \_Traits >, 2371
- ~basic\_ostringstream
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2415
- ~basic\_regex
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 2456
- ~basic\_streambuf
  - std::basic\_streambuf< \_CharT, \_Traits >, 2468
- ~basic\_string
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2495
- ~basic\_stringstream

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2586
- ~collate
  - std::collate< \_CharT >, 2730
- ~ctype
  - std::ctype< char >, 2775
  - std::ctype< wchar\_t >, 2792
- ~deque
  - std::deque< \_Tp, \_Alloc >, 2860
- ~facet
  - std::locale::facet, 3200
- ~forward\_list
  - std::forward\_list< \_Tp, \_Alloc >, 2949
- ~gslice
  - Numeric Arrays, 250
- ~ios\_base
  - std::ios\_base, 3062
- ~list
  - std::list< \_Tp, \_Alloc >, 3164
- ~locale
  - std::locale, 3190
- ~map
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3224
- ~match\_results
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3257
- ~messages
  - std::messages< \_CharT >, 3279
- ~money\_get
  - std::money\_get< \_CharT, \_InIter >, 3292
- ~money\_put
  - std::money\_put< \_CharT, \_OutIter >, 3298
- ~moneypunct
  - std::moneypunct< \_CharT, \_Intl >, 3305
- ~multimap
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3335
- ~multiset
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3368
- ~num\_get
  - std::num\_get< \_CharT, \_InIter >, 3407
- ~num\_put
  - std::num\_put< \_CharT, \_OutIter >, 3429
- ~numpunct
  - std::numpunct< \_CharT >, 3472
- ~sentry
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 2406
- ~set
  - std::set< \_Key, \_Compare, \_Alloc >, 3597
- ~stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1046
- ~temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 1097
- ~time\_get
  - std::time\_get< \_CharT, \_InIter >, 3671
- ~time\_put
  - std::time\_put< \_CharT, \_OutIter >, 3698
- ~type\_info
  - std::type\_info, 3749
- ~unique\_ptr
  - std::unique\_ptr< \_Tp, \_Dp >, 3766
  - std::unique\_ptr< \_Tp[], \_Dp >, 3773
- ~vector
  - std::vector< \_Tp, \_Alloc >, 3911
- a
  - std::extreme\_value\_distribution< \_RealType >, 2933
  - std::weibull\_distribution< \_RealType >, 3958
- a\_const\_iterator
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Ltr, \_ATraits, \_Alloc >, 1621
- abi, 562
- abs
  - Complex Numbers, 56
- access\_traits
  - \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >, 1614
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Ltr, \_ATraits, \_Alloc >, 1622
- accumulate
  - std, 687, 688
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1307
- acos
  - std, 689
- acosh
  - std, 689
- Adaptors for pointers to functions, 5
  - ptr\_fun, 6
- Adaptors for pointers to members, 7
- add\_const\_t
  - Metaprogramming, 177
- add\_cv\_t
  - Metaprogramming, 177
- add\_lvalue\_reference\_t
  - Metaprogramming, 177
- add\_pointer\_t
  - Metaprogramming, 177
- add\_rvalue\_reference\_t
  - Metaprogramming, 177
- add\_volatile\_t
  - Metaprogramming, 178
- addressof
  - Utilities, 438
- adjacent\_difference
  - std, 689, 690
- adjacent\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1307

- adjacent\_find
  - Non-Mutating, [213](#), [214](#)
- adjustfield
  - std::basic\_fstream< \_CharT, \_Traits >, [2052](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2106](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2138](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2203](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2255](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2311](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2355](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2398](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2443](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2633](#)
  - std::ios\_base, [3069](#)
- adopt\_lock
  - Mutexes, [209](#)
- advance
  - std, [690](#)
- airy\_ai
  - \_\_gnu\_cxx, [463](#)
- airy\_aif
  - \_\_gnu\_cxx, [463](#)
- airy\_ail
  - \_\_gnu\_cxx, [463](#)
- airy\_bi
  - \_\_gnu\_cxx, [464](#)
- airy\_bif
  - \_\_gnu\_cxx, [464](#)
- airy\_bil
  - \_\_gnu\_cxx, [464](#)
- algo.h, [3967](#)
- algbase.h, [3976](#)
- algorithm, [3978](#), [3980](#)
- algorithmfwd.h, [3981](#), [3986](#)
- Algorithms, [8](#)
- align
  - std, [691](#)
- aligned\_buffer.h, [3995](#)
- aligned\_storage\_t
  - Metaprogramming, [178](#)
- alignment\_value
  - Metaprogramming, [182](#)
- all
  - std::bitset< \_Nb >, [2664](#)
  - std::locale, [3195](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3720](#)
- all\_of
  - Non-Mutating, [214](#)
- alloc\_traits.h, [3995](#), [3996](#)
- allocate
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [868–870](#)
  - std::allocator\_traits< \_Alloc >, [1912](#), [1913](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1919](#), [1920](#)
- allocate\_shared
  - Pointer Abstractions, [292](#)
  - std::shared\_ptr< \_Tp >, [3635](#)
- allocated\_ptr.h, [3996](#)
- allocator.h, [3997](#)
- allocator\_type
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [1622](#)
  - std::allocator\_traits< \_Alloc >, [1909](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1917](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3590](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3781](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3814](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3844](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3874](#)
- Allocators, [9](#)
  - \_\_allocator\_base, [10](#)
- alpha
  - std::gamma\_distribution< \_RealType >, [2995](#)
- any, [3997](#)
  - std::bitset< \_Nb >, [2664](#)
  - std::experimental::fundamentals\_v1::any, [2910](#), [2911](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3720](#)
- any\_cast
  - Type-safe container of any type, [425–427](#)
- any\_of
  - Non-Mutating, [215](#)
- app
  - std::basic\_fstream< \_CharT, \_Traits >, [2052](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2106](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2138](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2203](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2256](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2312](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2356](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2398](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2443](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2633](#)
  - std::ios\_base, [3069](#)
- append
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [898–901](#)

- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1177](#), [1178](#)
- `std::basic_string< _CharT, _Traits, _Alloc >`, [2495–2498](#)
- `std::tr2::dynamic_bitset< _WordT, _Alloc >`, [3720](#), [3721](#)
- `apply`
  - Numeric Arrays, [250](#), [251](#)
- `apply_generator`
  - `__gnu_cxx::typelist`, [482](#)
- `arg`
  - Complex Numbers, [56](#)
  - `std`, [692](#)
- `argument_type`
  - `__gnu_cxx::detail::Ffit_finder< _Tp >`, [877](#)
  - `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`, [971](#)
  - `__gnu_cxx::select1st< _Pair >`, [1037](#)
  - `__gnu_cxx::select2nd< _Pair >`, [1038](#)
  - `__gnu_cxx::subtractive_rng`, [1094](#)
  - `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`, [1108](#)
  - `__gnu_parallel::binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, [1207](#)
  - `__gnu_parallel::binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, [1209](#)
  - `__gnu_parallel::unary_negate< _Predicate, argument_type >`, [1243](#)
  - `std::maybe_unary_or_binary_function< _Res, _T1 >`, [1883](#)
  - `std::binder1st< _Operation >`, [2650](#)
  - `std::binder2nd< _Operation >`, [2651](#)
  - `std::const_mem_fun_ref_t< _Ret, _Tp >`, [2751](#)
  - `std::const_mem_fun_t< _Ret, _Tp >`, [2753](#)
  - `std::hash< __gnu_cxx::throw_value_limit >`, [3016](#)
  - `std::hash< __gnu_cxx::throw_value_random >`, [3018](#)
  - `std::logical_not< _Tp >`, [3207](#)
  - `std::mem_fun_ref_t< _Ret, _Tp >`, [3269](#)
  - `std::mem_fun_t< _Ret, _Tp >`, [3270](#)
  - `std::negate< _Tp >`, [3388](#)
  - `std::pointer_to_unary_function< _Arg, _Result >`, [3522](#)
  - `std::unary_function< _Arg, _Result >`, [3750](#)
  - `std::unary_negate< _Predicate >`, [3751](#)
- Arithmetic Classes, [11](#)
- `array`, [3998–4000](#)
- Array creation functions, [12](#)
- `array_allocator.h`, [4001](#)
- `asin`
  - `std`, [692](#)
- `asinh`
- `std`, [692](#)
- `assertions.h`, [4001](#)
- `assign`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [902–906](#)
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1179](#), [1180](#)
  - `std::basic_regex< _Ch_type, _Rx_traits >`, [2456–2459](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [2499–2503](#)
  - `std::deque< _Tp, _Alloc >`, [2866](#), [2867](#)
  - `std::forward_list< _Tp, _Alloc >`, [2949](#), [2950](#)
  - `std::list< _Tp, _Alloc >`, [3164](#), [3165](#)
  - `std::vector< _Tp, _Alloc >`, [3911](#), [3912](#)
- `assoc_container.hpp`, [4002](#)
- `assoc_laguerre`
  - Mathematical Special Functions, [135](#), [166](#)
- `assoc_laguerref`
  - Mathematical Special Functions, [136](#)
- `assoc_laguerrel`
  - Mathematical Special Functions, [136](#)
- `assoc_legendre`
  - Mathematical Special Functions, [136](#), [166](#)
- `assoc_legendref`
  - Mathematical Special Functions, [137](#)
- `assoc_legendrel`
  - Mathematical Special Functions, [138](#)
- Associative, [13](#)
- `async`
  - Futures, [99](#)
- `at`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [907](#)
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1180](#), [1181](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [2503](#), [2504](#)
  - `std::deque< _Tp, _Alloc >`, [2867](#), [2868](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [3224](#)
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3788](#)
  - `std::vector< _Tp, _Alloc >`, [3913](#)
- `atan`
  - `std`, [692](#)
- `atanh`
  - `std`, [692](#)
- `ate`
  - `std::basic_fstream< _CharT, _Traits >`, [2052](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [2107](#)
  - `std::basic_ios< _CharT, _Traits >`, [2139](#)
  - `std::basic_iostream< _CharT, _Traits >`, [2203](#)
  - `std::basic_istream< _CharT, _Traits >`, [2256](#)

- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, [2312](#)
- std::basic\_ofstream< \_CharT, \_Traits >, [2356](#)
- std::basic\_ostream< \_CharT, \_Traits >, [2398](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2443](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2634](#)
- std::ios\_base, [3069](#)
- atomic, [4003](#)
- atomic\_base.h, [4007](#)
- atomic\_bool
  - Atomics, [19](#)
- ATOMIC\_BOOL\_LOCK\_FREE
  - Atomics, [18](#)
- atomic\_char
  - Atomics, [19](#)
- atomic\_char16\_t
  - Atomics, [19](#)
- atomic\_char32\_t
  - Atomics, [19](#)
- atomic\_compare\_exchange\_strong
  - Pointer Abstractions, [292](#), [293](#)
- atomic\_compare\_exchange\_strong\_explicit
  - Pointer Abstractions, [293](#), [295](#)
- atomic\_compare\_exchange\_weak
  - Pointer Abstractions, [296](#)
- atomic\_compare\_exchange\_weak\_explicit
  - Pointer Abstractions, [297](#)
- atomic\_exchange
  - Pointer Abstractions, [298](#), [299](#)
- atomic\_exchange\_explicit
  - Pointer Abstractions, [299](#), [300](#)
- atomic\_futex.h, [4008](#)
- atomic\_int
  - Atomics, [20](#)
- atomic\_int16\_t
  - Atomics, [20](#)
- atomic\_int32\_t
  - Atomics, [20](#)
- atomic\_int64\_t
  - Atomics, [20](#)
- atomic\_int8\_t
  - Atomics, [20](#)
- atomic\_int\_fast16\_t
  - Atomics, [21](#)
- atomic\_int\_fast32\_t
  - Atomics, [21](#)
- atomic\_int\_fast64\_t
  - Atomics, [21](#)
- atomic\_int\_fast8\_t
  - Atomics, [21](#)
- atomic\_int\_least16\_t
  - Atomics, [21](#)
- atomic\_int\_least32\_t
  - Atomics, [22](#)
- atomic\_int\_least64\_t
  - Atomics, [22](#)
- atomic\_int\_least8\_t
  - Atomics, [22](#)
- atomic\_intmax\_t
  - Atomics, [22](#)
- atomic\_intptr\_t
  - Atomics, [22](#)
- atomic\_is\_lock\_free
  - Pointer Abstractions, [300](#), [301](#)
- atomic\_llong
  - Atomics, [23](#)
- atomic\_load
  - Pointer Abstractions, [301](#), [302](#)
- atomic\_load\_explicit
  - Pointer Abstractions, [302](#), [303](#)
- atomic\_lockfree\_defines.h, [4008](#)
- atomic\_long
  - Atomics, [23](#)
- atomic\_ptrdiff\_t
  - Atomics, [23](#)
- atomic\_schar
  - Atomics, [23](#)
- atomic\_short
  - Atomics, [23](#)
- atomic\_size\_t
  - Atomics, [24](#)
- atomic\_store
  - Pointer Abstractions, [303](#), [304](#)
- atomic\_store\_explicit
  - Pointer Abstractions, [304](#), [305](#)
- atomic\_uchar
  - Atomics, [24](#)
- atomic\_uint
  - Atomics, [24](#)
- atomic\_uint16\_t
  - Atomics, [24](#)
- atomic\_uint32\_t
  - Atomics, [24](#)
- atomic\_uint64\_t
  - Atomics, [25](#)
- atomic\_uint8\_t
  - Atomics, [25](#)
- atomic\_uint\_fast16\_t
  - Atomics, [25](#)
- atomic\_uint\_fast32\_t
  - Atomics, [25](#)
- atomic\_uint\_fast64\_t
  - Atomics, [25](#)
- atomic\_uint\_fast8\_t
  - Atomics, [26](#)
- atomic\_uint\_least16\_t

- Atomics, [26](#)
- atomic\_uint\_least32\_t
  - Atomics, [26](#)
- atomic\_uint\_least64\_t
  - Atomics, [26](#)
- atomic\_uint\_least8\_t
  - Atomics, [26](#)
- atomic\_uintmax\_t
  - Atomics, [27](#)
- atomic\_uintptr\_t
  - Atomics, [27](#)
- atomic\_ullong
  - Atomics, [27](#)
- atomic\_ulong
  - Atomics, [27](#)
- atomic\_ushort
  - Atomics, [27](#)
- atomic\_wchar\_t
  - Atomics, [28](#)
- atomic\_word.h, [4008](#)
- atomicity.h, [4009](#)
- Atomics, [14](#)
  - atomic\_bool, [19](#)
  - ATOMIC\_BOOL\_LOCK\_FREE, [18](#)
  - atomic\_char, [19](#)
  - atomic\_char16\_t, [19](#)
  - atomic\_char32\_t, [19](#)
  - atomic\_int, [20](#)
  - atomic\_int16\_t, [20](#)
  - atomic\_int32\_t, [20](#)
  - atomic\_int64\_t, [20](#)
  - atomic\_int8\_t, [20](#)
  - atomic\_int\_fast16\_t, [21](#)
  - atomic\_int\_fast32\_t, [21](#)
  - atomic\_int\_fast64\_t, [21](#)
  - atomic\_int\_fast8\_t, [21](#)
  - atomic\_int\_least16\_t, [21](#)
  - atomic\_int\_least32\_t, [22](#)
  - atomic\_int\_least64\_t, [22](#)
  - atomic\_int\_least8\_t, [22](#)
  - atomic\_intmax\_t, [22](#)
  - atomic\_intptr\_t, [22](#)
  - atomic\_llong, [23](#)
  - atomic\_long, [23](#)
  - atomic\_ptrdiff\_t, [23](#)
  - atomic\_schar, [23](#)
  - atomic\_short, [23](#)
  - atomic\_size\_t, [24](#)
  - atomic\_uchar, [24](#)
  - atomic\_uint, [24](#)
  - atomic\_uint16\_t, [24](#)
  - atomic\_uint32\_t, [24](#)
  - atomic\_uint64\_t, [25](#)
  - atomic\_uint8\_t, [25](#)
  - atomic\_uint\_fast16\_t, [25](#)
  - atomic\_uint\_fast32\_t, [25](#)
  - atomic\_uint\_fast64\_t, [25](#)
  - atomic\_uint\_fast8\_t, [26](#)
  - atomic\_uint\_least16\_t, [26](#)
  - atomic\_uint\_least32\_t, [26](#)
  - atomic\_uint\_least64\_t, [26](#)
  - atomic\_uint\_least8\_t, [26](#)
  - atomic\_uintmax\_t, [27](#)
  - atomic\_uintptr\_t, [27](#)
  - atomic\_ullong, [27](#)
  - atomic\_ulong, [27](#)
  - atomic\_ushort, [27](#)
  - atomic\_wchar\_t, [28](#)
  - kill\_dependency, [28](#)
  - memory\_order, [28](#)
- auto\_ptr
  - std::auto\_ptr< \_Tp >, [1949](#), [1950](#)
- auto\_ptr.h, [4009](#)
- awk
  - std::regex\_constants, [849](#)
- b
  - std::extreme\_value\_distribution< \_RealType >, [2933](#)
  - std::weibull\_distribution< \_RealType >, [3958](#)
- back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [908](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1181](#), [1182](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2504](#), [2505](#)
  - std::deque< \_Tp, \_Alloc >, [2868](#), [2869](#)
  - std::list< \_Tp, \_Alloc >, [3166](#)
  - std::queue< \_Tp, \_Sequence >, [3540](#)
  - std::vector< \_Tp, \_Alloc >, [3915](#)
- back\_insert\_iterator
  - std::back\_insert\_iterator< \_Container >, [1956](#)
- back\_inserter
  - Iterators, [126](#)
- backward\_warning.h, [4010](#)
- bad
  - std::basic\_fstream< \_CharT, \_Traits >, [2005](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2070](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2123](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2157](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2220](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2275](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2329](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2372](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2417](#)

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2587
- badbit
  - std::basic\_fstream< \_CharT, \_Traits >, 2052
  - std::basic\_ifstream< \_CharT, \_Traits >, 2107
  - std::basic\_ios< \_CharT, \_Traits >, 2139
  - std::basic\_iostream< \_CharT, \_Traits >, 2203
  - std::basic\_istream< \_CharT, \_Traits >, 2256
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2312
  - std::basic\_ofstream< \_CharT, \_Traits >, 2356
  - std::basic\_ostream< \_CharT, \_Traits >, 2398
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2443
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2634
  - std::ios\_base, 3069
- balanced\_quicksort.h, 4010
- base
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, 1123
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, 1141
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2885
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 3042
  - std::reverse\_iterator< \_Iterator >, 3578
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3640
- Base and Implementation Classes, 29, 31
  - \_Opcode, 30
  - \_\_clp2, 33
- Base and Policy Classes, 34–36
- base.h, 4011
- basefield
  - std::basic\_fstream< \_CharT, \_Traits >, 2053
  - std::basic\_ifstream< \_CharT, \_Traits >, 2107
  - std::basic\_ios< \_CharT, \_Traits >, 2139
  - std::basic\_iostream< \_CharT, \_Traits >, 2204
  - std::basic\_istream< \_CharT, \_Traits >, 2256
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2312
  - std::basic\_ofstream< \_CharT, \_Traits >, 2356
  - std::basic\_ostream< \_CharT, \_Traits >, 2398
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2444
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2634
  - std::ios\_base, 3069
- basic
  - std::regex\_constants, 849
- basic\_file.h, 4012
- basic\_filebuf
  - std::basic\_filebuf< \_CharT, \_Traits >, 1968
- basic\_fstream
  - std::basic\_fstream< \_CharT, \_Traits >, 2003
- basic\_ifstream
  - std::basic\_ifstream< \_CharT, \_Traits >, 2069
- basic\_ios
  - std::basic\_ios< \_CharT, \_Traits >, 2122, 2123
- basic\_ios.h, 4012
- basic\_ios.tcc, 4013
- basic\_iostream
  - std::basic\_iostream< \_CharT, \_Traits >, 2156
- basic\_istream
  - std::basic\_istream< \_CharT, \_Traits >, 2219
- basic\_istreamstream
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2274
- basic\_iterator.h, 4013
- basic\_ofstream
  - std::basic\_ofstream< \_CharT, \_Traits >, 2327
- basic\_ostream
  - std::basic\_ostream< \_CharT, \_Traits >, 2371
- basic\_ostringstream
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2415
- basic\_regex
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 2452–2455
- basic\_streambuf
  - std::basic\_streambuf< \_CharT, \_Traits >, 2468
- basic\_string
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2490–2495
- basic\_string.h, 4013
- basic\_string.tcc, 4016
- basic\_stringbuf
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2554, 2555
- basic\_stringstream
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2585, 2586
- before\_begin
  - std::forward\_list< \_Tp, \_Alloc >, 2950, 2951
- beg
  - std::basic\_fstream< \_CharT, \_Traits >, 2053
  - std::basic\_ifstream< \_CharT, \_Traits >, 2107
  - std::basic\_ios< \_CharT, \_Traits >, 2139
  - std::basic\_iostream< \_CharT, \_Traits >, 2204
  - std::basic\_istream< \_CharT, \_Traits >, 2256
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2312
  - std::basic\_ofstream< \_CharT, \_Traits >, 2356
  - std::basic\_ostream< \_CharT, \_Traits >, 2398
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2444



- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2634
- std::ios\_base, 3070
- begin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 908, 909
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 1097
  - \_\_gnu\_parallel::PseudoSequence< \_Tp, \_DifferenceTp >, 1296
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1600
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 1625
- Numeric Arrays, 251, 252
- std, 693, 695
- std::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1891
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2505
- std::deque< \_Tp, \_Alloc >, 2869
- std::forward\_list< \_Tp, \_Alloc >, 2951
- std::list< \_Tp, \_Alloc >, 3166, 3167
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3225
- std::match\_results< \_Bi\_iter, \_Alloc >, 3257
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3335
- std::multiset< \_Key, \_Compare, \_Alloc >, 3368
- std::set< \_Key, \_Compare, \_Alloc >, 3598
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3789, 3790
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3821, 3822
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3850, 3851
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3880, 3881
- std::vector< \_Tp, \_Alloc >, 3915, 3916
- Bernoulli Distributions, 37
  - operator!=, 37, 38
  - operator<<, 38, 39
  - operator>>, 39, 40
- bernoulli\_distribution
  - std::bernoulli\_distribution, 2642
- beta
  - Mathematical Special Functions, 138, 166
  - std::gamma\_distribution< \_RealType >, 2995
- betaf
  - Mathematical Special Functions, 139
- betal
  - Mathematical Special Functions, 139
- bin\_search\_tree.hpp, 4017
- binary
  - std::basic\_fstream< \_CharT, \_Traits >, 2053
  - std::basic\_ifstream< \_CharT, \_Traits >, 2107
- std::basic\_ios< \_CharT, \_Traits >, 2139
- std::basic\_iostream< \_CharT, \_Traits >, 2204
- std::basic\_istream< \_CharT, \_Traits >, 2257
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2313
- std::basic\_ofstream< \_CharT, \_Traits >, 2357
- std::basic\_ostream< \_CharT, \_Traits >, 2399
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2444
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2634
- std::ios\_base, 3070
- Binary Search, 41
  - binary\_search, 42
  - equal\_range, 43
  - lower\_bound, 44
  - upper\_bound, 45
- binary\_heap.hpp, 4017
- binary\_heap\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1387
- binary\_heap\_point\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1392, 1393
- binary\_search
  - Binary Search, 42
- bind
  - Binder Classes, 48
- bind1st
  - Binder Classes, 48
- bind2nd
  - Binder Classes, 49
- Binder Classes, 47
  - bind, 48
  - bind1st, 48
  - bind2nd, 49
- binders.h, 4018
- binomial\_heap.hpp, 4018
- binomial\_heap\_base.hpp, 4019
- bitmap\_allocator.h, 4019
  - \_BALLOC\_ALIGN\_BYTES, 4020
- bitset, 4021, 4022
  - std::bitset< \_Nb >, 2662, 2663
- bool\_set, 4023
  - std::tr2::bool\_set, 3711
- bool\_set.tcc, 4024
- boolalpha
  - std, 695
  - std::basic\_fstream< \_CharT, \_Traits >, 2053
  - std::basic\_ifstream< \_CharT, \_Traits >, 2108
  - std::basic\_ios< \_CharT, \_Traits >, 2140
  - std::basic\_iostream< \_CharT, \_Traits >, 2204
  - std::basic\_istream< \_CharT, \_Traits >, 2257



- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 2313
- std::basic\_ofstream< \_CharT, \_Traits >, 2357
- std::basic\_ostream< \_CharT, \_Traits >, 2399
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2444
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2635
- std::ios\_base, 3070
- Boolean Operations Classes, 50
- boost\_concept\_check.h, 4024
- Branch-Based, 51
- branch\_policy.hpp, 4025
- bucket
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, 1141
- bucket\_count
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3790
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3822
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3852
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3882
- c
  - std::queue< \_Tp, \_Sequence >, 3542
- c++0x\_warning.h, 4025
- c++allocator.h, 4026
- c++config.h, 4026
- c++io.h, 4032
- c++locale.h, 4032
- c++locale\_internal.h, 4033
- c\_str
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 909
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2505
- cache\_line\_size
  - \_\_gnu\_parallel::Settings, 1307
- call\_once
  - std, 695
  - std::once\_flag, 3484
- capacity
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 909
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1182
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2505
  - std::vector< \_Tp, \_Alloc >, 3916
- cassert, 4033
- cast.h, 4033
- category
  - std::locale, 3187
- cbefore\_begin
  - std::forward\_list< \_Tp, \_Alloc >, 2951
- cbegin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 909
  - std, 695
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2506
  - std::deque< \_Tp, \_Alloc >, 2869
  - std::forward\_list< \_Tp, \_Alloc >, 2952
  - std::list< \_Tp, \_Alloc >, 3167
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3225
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3257
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3336
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3368
  - std::set< \_Key, \_Compare, \_Alloc >, 3598
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3790
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3822
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3852
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3882
  - std::vector< \_Tp, \_Alloc >, 3916
- cc\_hash\_max\_collision\_check\_resize\_trigger
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1348
- cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp, 4034
- cc\_hash\_table
  - \_\_gnu\_pbds::cc\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, 1354–1357
- cc\_ht\_map.hpp, 4034
- ccomplex, 4035
- cctype, 4035, 4036
- cend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 910
  - std, 696
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2506
  - std::deque< \_Tp, \_Alloc >, 2869
  - std::forward\_list< \_Tp, \_Alloc >, 2952
  - std::list< \_Tp, \_Alloc >, 3167
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3225
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3258
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3336
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3368
  - std::set< \_Key, \_Compare, \_Alloc >, 3598
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3791
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,

- [\\_Alloc >, 3823](#)
  - [std::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3853](#)
  - [std::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3883](#)
  - [std::vector< \\_Tp, \\_Alloc >, 3916](#)
- cerr
  - [std, 789](#)
- cerrno, [4036](#)
- cfenv, [4036](#), [4037](#)
- cfloat, [4037](#)
- char\_traits.h, [4038](#)
- char\_type
  - [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >, 1682](#)
  - [std::basic\\_ios< \\_CharT, \\_Traits >, 2118](#)
  - [std::basic\\_streambuf< \\_CharT, \\_Traits >, 2467](#)
  - [std::collate< \\_CharT >, 2729](#)
  - [std::collate\\_byname< \\_CharT >, 2736](#)
  - [std::ctype< char >, 2774](#)
  - [std::ctype< wchar\\_t >, 2791](#)
  - [std::ctype\\_byname< char >, 2827](#)
  - [std::istreambuf\\_iterator< \\_CharT, \\_Traits >, 3137](#)
  - [std::messages< \\_CharT >, 3278](#)
  - [std::money\\_get< \\_CharT, \\_InIter >, 3291](#)
  - [std::money\\_put< \\_CharT, \\_OutIter >, 3297](#)
  - [std::moneypunct< \\_CharT, \\_Intl >, 3304](#)
  - [std::num\\_get< \\_CharT, \\_InIter >, 3406](#)
  - [std::num\\_put< \\_CharT, \\_OutIter >, 3427](#)
  - [std::numpunct< \\_CharT >, 3470](#)
  - [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >, 3486](#)
  - [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >, 3491](#)
  - [std::time\\_get< \\_CharT, \\_InIter >, 3671](#)
  - [std::time\\_put< \\_CharT, \\_OutIter >, 3697](#)
  - [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >, 3937](#)
- checkers.h, [4038](#)
- chrono, [4039](#), [4042](#)
  - [high\\_resolution\\_clock, 4042](#)
- cin
  - [std, 790](#)
- cinttypes, [4042](#), [4043](#)
- ciso646, [4043](#)
- classic
  - [std::locale, 3191](#)
- classic\_table
  - [std::ctype< char >, 2775](#)
  - [std::ctype\\_byname< char >, 2827](#)
- clear
  - [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >, 910](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >, 2005](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >, 2071](#)
  - [std::basic\\_ios< \\_CharT, \\_Traits >, 2124](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >, 2157](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >, 2220](#)
- [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >, 2275](#)
- [std::basic\\_ofstream< \\_CharT, \\_Traits >, 2329](#)
- [std::basic\\_ostream< \\_CharT, \\_Traits >, 2372](#)
- [std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >, 2417](#)
- [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >, 2506](#)
- [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >, 2588](#)
- [std::deque< \\_Tp, \\_Alloc >, 2870](#)
- [std::experimental::fundamentals\\_v1::any, 2912](#)
- [std::forward\\_list< \\_Tp, \\_Alloc >, 2952](#)
- [std::list< \\_Tp, \\_Alloc >, 3167](#)
- [std::map< \\_Key, \\_Tp, \\_Compare, \\_Alloc >, 3226](#)
- [std::multimap< \\_Key, \\_Tp, \\_Compare, \\_Alloc >, 3336](#)
- [std::multiset< \\_Key, \\_Compare, \\_Alloc >, 3369](#)
- [std::set< \\_Key, \\_Compare, \\_Alloc >, 3598](#)
- [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >, 3721](#)
- [std::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 3792](#)
- [std::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 3824](#)
- [std::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3853](#)
- [std::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3883](#)
- [std::vector< \\_Tp, \\_Alloc >, 3917](#)
- climits, [4043](#), [4044](#)
- locale, [4044](#)
- clog
  - [std, 790](#)
- close
  - [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >, 984](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >, 1047](#)
  - [std::basic\\_filebuf< \\_CharT, \\_Traits >, 1969](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >, 2005](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >, 2071](#)
  - [std::basic\\_ofstream< \\_CharT, \\_Traits >, 2329](#)
- cmath, [4044](#), [4047](#)
- cmp\_fn
  - [\\_\\_gnu\\_pbds::tree< Key, Mapped, Cmp\\_Fn, Tag, Node\\_Update, \\_Alloc >, 1608](#)
- cmp\_fn\_imps.hpp, [4049](#)
- code
  - [std::regex\\_error, 3555](#)
- codecvt, [4049](#)
- codecvt.h, [4050](#)
- codecvt\_specializations.h, [4050](#)
- collate
  - [std::collate< \\_CharT >, 2729](#), [2730](#)
  - [std::locale, 3195](#)
  - [std::regex\\_constants, 849](#)
- combine

- std::locale, 3191
- common\_type\_t
  - Metaprogramming, 178
- comp\_ellint\_1
  - Mathematical Special Functions, 139, 166
- comp\_ellint\_1f
  - Mathematical Special Functions, 140
- comp\_ellint\_1l
  - Mathematical Special Functions, 140
- comp\_ellint\_2
  - Mathematical Special Functions, 141, 167
- comp\_ellint\_2f
  - Mathematical Special Functions, 142
- comp\_ellint\_2l
  - Mathematical Special Functions, 142
- comp\_ellint\_3
  - Mathematical Special Functions, 142, 167
- comp\_ellint\_3f
  - Mathematical Special Functions, 143
- comp\_ellint\_3l
  - Mathematical Special Functions, 143
- compare
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 910–913
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1182, 1183
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2506–2509
  - std::collate< \_CharT >, 2730
  - std::collate\_byname< \_CharT >, 2736
  - std::sub\_match< \_Biter >, 3656, 3657
- Comparison Classes, 52
- compatibility.h, 4051
- compiletime\_settings.h, 4052
  - \_GLIBCXX\_CALL, 4052
  - \_GLIBCXX\_PARALLEL\_ASSERTIONS, 4052
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_L1, 4052
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_TLB, 4053
  - \_GLIBCXX\_SCALE\_DOWN\_FPU, 4053
  - \_GLIBCXX\_VERBOSE\_LEVEL, 4053
- complex, 4053, 4058
  - std::complex< \_Tp >, 2742
- Complex Numbers, 53
  - abs, 56
  - arg, 56
  - conj, 57
  - cos, 57
  - cosh, 57
  - exp, 57
  - fabs, 58
  - log, 58
  - log10, 58
  - norm, 58
  - operator!=, 59
  - operator<<, 64
  - operator>>, 66
  - operator\*, 59, 60
  - operator\*==, 60
  - operator+, 61
  - operator+==, 62
  - operator-, 62, 63
  - operator-=, 63
  - operator/, 63, 64
  - operator/=, 64
  - operator=, 65
  - operator==, 65, 66
  - polar, 66
  - pow, 66, 67
  - sin, 68
  - sinh, 68
  - sqrt, 68
  - tan, 68
  - tanh, 69
- complex.h, 4059
- compose1
  - SGI, 377
- compose2
  - SGI, 378
- concept\_check.h, 4059
- concurrency.h, 4059
- Concurrency, 70
- cond\_dealtor.hpp, 4060
- cond\_key\_dtor\_entry\_dealtor.hpp, 4060
- Condition Variables, 71
  - cv\_status, 71
- condition\_variable, 4060
- conditional\_t
  - Metaprogramming, 178
- conf\_hyperg
  - \_\_gnu\_cxx, 464
  - std::tr1, 861
- conf\_hypergf
  - \_\_gnu\_cxx, 465
- conf\_hypergl
  - \_\_gnu\_cxx, 465
- conj
  - Complex Numbers, 57
- Const-propagating wrapper, 72
- const\_iterator
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 1625
  - std::set< \_Key, \_Compare, \_Alloc >, 3590
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3781

- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3814](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3844](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3874](#)
- const\_iterator.hpp, [4061](#), [4062](#)
- const\_iterator\_, [1647](#)
  - const\_iterator\_, [1650](#)
  - const\_pointer, [1648](#)
  - const\_reference, [1648](#)
  - difference\_type, [1648](#)
  - iterator\_category, [1649](#)
  - m\_p\_tbl, [1652](#)
  - operator!=, [1650](#)
  - operator\*, [1650](#)
  - operator++, [1651](#)
  - operator->, [1651](#)
  - operator==, [1651](#)
  - pointer, [1649](#)
  - reference, [1649](#)
  - value\_type, [1649](#)
- const\_local\_iterator
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3781](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3814](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3844](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3874](#)
- const\_pointer
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1385](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1391](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [1442](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [1448](#)
- const\_iterator\_, [1648](#)
- iterator\_, [1654](#)
- point\_const\_iterator\_, [1659](#)
- point\_iterator\_, [1664](#)
- std::allocator\_traits< \_Alloc >, [1910](#)
- std::allocator\_traits< allocator< \_Tp > >, [1917](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3590](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3782](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3815](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3844](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3874](#)
- >, [3874](#)
- const\_pointer\_cast
  - std, [696](#)
- const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [1371](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [1377](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1386](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1391](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [1442](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [1448](#)
- const\_iterator\_, [1648](#)
- iterator\_, [1654](#)
- point\_const\_iterator\_, [1659](#)
- point\_iterator\_, [1664](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3590](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3782](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3815](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3845](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3875](#)
- const\_reverse\_iterator
  - std::set< \_Key, \_Compare, \_Alloc >, [3591](#)
- const\_void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [867](#)
  - std::allocator\_traits< \_Alloc >, [1910](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1917](#)
- constant0
  - SGI, [378](#)
- constant1
  - SGI, [378](#)
- constant2
  - SGI, [378](#)
- construct
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [870](#), [871](#)
  - std::allocator\_traits< \_Alloc >, [1913](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1920](#)
- constructor\_destructor\_fn\_imps.hpp, [4062](#), [4063](#)
- constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp, [4063](#)
- constructor\_destructor\_store\_hash\_fn\_imps.hpp, [4063](#)
- constructors\_destructor\_fn\_imps.hpp, [4063](#)–[4065](#)
- container\_base\_dispatch.hpp, [4065](#)
- container\_type
  - std::back\_insert\_iterator< \_Container >, [1955](#)

- std::front\_insert\_iterator< \_Container >, [2971](#)
- std::insert\_iterator< \_Container >, [3050](#)
- Containers, [74](#), [75](#)
- converted
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, [3964](#)
- copy
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [914](#)
  - Mutating, [185](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2510](#)
- copy\_backward
  - Mutating, [185](#)
- copy\_if
  - Mutating, [186](#)
- copy\_n
  - Mutating, [186](#)
  - SGL, [379](#)
- copy\_options
  - Filesystem TS, [94](#)
- copyfmt
  - std::basic\_fstream< \_CharT, \_Traits >, [2006](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2071](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2124](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2157](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2221](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2276](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2330](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2372](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2417](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2588](#)
- cos
  - Complex Numbers, [57](#)
- cosh
  - Complex Numbers, [57](#)
- count
  - Non-Mutating, [216](#)
  - std::bitset< \_Nb >, [2664](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3226](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3336](#), [3337](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3369](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3598](#), [3599](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3721](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3792](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3824](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3854](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3884](#)
- count\_if
  - Non-Mutating, [216](#)
- count\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, [1308](#)
- cout
  - std, [790](#)
- cpp\_type\_traits.h, [4066](#)
- cpu\_defines.h, [4067](#)
- crbegin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [915](#)
  - std, [696](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2511](#)
  - std::deque< \_Tp, \_Alloc >, [2870](#)
  - std::list< \_Tp, \_Alloc >, [3168](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3227](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3337](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3370](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3599](#)
  - std::vector< \_Tp, \_Alloc >, [3917](#)
- cref
  - std, [697](#)
- cregex\_token\_iterator
  - Regular Expressions, [331](#)
- crend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [915](#)
  - std, [697](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2511](#)
  - std::deque< \_Tp, \_Alloc >, [2870](#)
  - std::list< \_Tp, \_Alloc >, [3168](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3227](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3337](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3370](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3599](#)
  - std::vector< \_Tp, \_Alloc >, [3917](#)
- csetjmp, [4067](#)
- cshift
  - Numeric Arrays, [252](#)
- csignal, [4067](#)
- cstdalign, [4068](#)
- cstdarg, [4068](#)
- cstdbool, [4069](#)
- cstddef, [4069](#)
- cstdint, [4069](#), [4070](#)
- cstdio, [4070](#), [4071](#)
- cstdlib, [4071](#)
- cstring, [4072](#)
- csub\_match
  - Regular Expressions, [331](#)

- ctgmath, [4072](#), [4073](#)
- ctime, [4073](#)
- ctype
  - std::ctype< char >, [2774](#), [2775](#)
  - std::ctype< wchar\_t >, [2791](#), [2792](#)
  - std::locale, [3195](#)
- ctype\_base.h, [4073](#)
- ctype\_inline.h, [4074](#)
- cuchar, [4074](#)
- cur
  - std::basic\_fstream< \_CharT, \_Traits >, [2054](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2108](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2140](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2205](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2257](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2313](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2357](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2399](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2445](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2635](#)
  - std::ios\_base, [3070](#)
- curr\_symbol
  - std::moneypunct< \_CharT, \_Intl >, [3306](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3317](#)
- current\_exception
  - Exceptions, [86](#)
- cv\_status
  - Condition Variables, [71](#)
- cwchar, [4074](#), [4075](#)
- cwctype, [4075](#), [4076](#)
- cxxabi.h, [4076](#)
  - \_\_cxa\_demangle, [4078](#)
- cxxabi\_forced.h, [4079](#)
- cxxabi\_init\_exception.h, [4079](#)
- cxxabi\_tweaks.h, [4079](#)
- cyl\_bessel\_i
  - Mathematical Special Functions, [143](#), [167](#)
- cyl\_bessel\_if
  - Mathematical Special Functions, [144](#)
- cyl\_bessel\_il
  - Mathematical Special Functions, [144](#)
- cyl\_bessel\_j
  - Mathematical Special Functions, [145](#), [167](#)
- cyl\_bessel\_jf
  - Mathematical Special Functions, [145](#)
- cyl\_bessel\_jl
  - Mathematical Special Functions, [146](#)
- cyl\_bessel\_k
  - Mathematical Special Functions, [146](#), [168](#)
- cyl\_bessel\_kf
  - Mathematical Special Functions, [147](#)
- cyl\_bessel\_kl
  - Mathematical Special Functions, [147](#)
- cyl\_neumann
  - Mathematical Special Functions, [147](#), [168](#)
- cyl\_neumannf
  - Mathematical Special Functions, [148](#)
- cyl\_neumannl
  - Mathematical Special Functions, [148](#)
- data
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [915](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2511](#)
  - std::vector< \_Tp, \_Alloc >, [3917](#)
- Data Structure Type, [76](#)
- date\_order
  - std::time\_get< \_CharT, \_InIter >, [3672](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3685](#)
- deallocate
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [871](#), [872](#)
  - std::allocator\_traits< \_Alloc >, [1914](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1921](#)
- debug.h, [4080](#)
- debug\_allocator.h, [4081](#)
- debug\_fn\_imps.hpp, [4081](#)–[4083](#)
- debug\_map\_base.hpp, [4084](#)
- debug\_no\_store\_hash\_fn\_imps.hpp, [4084](#)
- debug\_store\_hash\_fn\_imps.hpp, [4084](#)
- dec
  - std, [698](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2054](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2108](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2140](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2205](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2257](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2313](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2357](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2400](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2445](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2635](#)
  - std::ios\_base, [3071](#)
- decay\_t
  - Metaprogramming, [178](#)
- decimal, [4084](#)
- Decimal Floating-Point Arithmetic, [77](#)
- decimal128
  - std::decimal::decimal128, [2843](#)
- decimal32
  - std::decimal::decimal32, [2845](#)
- decimal32\_to\_long\_long



- std::decimal, [838](#)
- decimal64
  - std::decimal::decimal64, [2847](#)
- decimal\_point
  - std::moneypunct< \_CharT, \_Intl >, [3306](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3318](#)
  - std::numpunct< \_CharT >, [3472](#)
  - std::numpunct\_byname< \_CharT >, [3479](#)
- default\_delete
  - std::default\_delete< \_Tp >, [2848](#)
  - std::default\_delete< \_Tp[] >, [2849](#)
- defaultfloat
  - std, [698](#)
- defer\_lock
  - Mutexes, [209](#)
- denorm\_absent
  - std, [673](#)
- denorm\_indeterminate
  - std, [673](#)
- denorm\_min
  - std::numeric\_limits< \_Tp >, [3444](#)
- denorm\_present
  - std, [673](#)
- densities
  - std::piecewise\_constant\_distribution< \_RealType >, [3507](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [3513](#)
- deque, [4094](#), [4095](#)
  - std::deque< \_Tp, \_Alloc >, [2856–2859](#)
- deque.tcc, [4096](#)
- destroy
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [872](#), [873](#)
  - std::allocator\_traits< \_Alloc >, [1914](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1921](#)
- Diagnostics, [78](#)
- difference\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [1371](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [1377](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1386](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1391](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< s, \_r >, [3662](#)
  - Node, \_Alloc >, [1442](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [1448](#)
  - const\_iterator\_, [1648](#)
  - iterator\_, [1654](#)
  - point\_const\_iterator\_, [1660](#)
  - point\_iterator\_, [1664](#)
- std::allocator\_traits< \_Alloc >, [1910](#)
- std::allocator\_traits< allocator< \_Tp > >, [1917](#)
- std::back\_insert\_iterator< \_Container >, [1955](#)
- std::front\_insert\_iterator< \_Container >, [2971](#)
- std::insert\_iterator< \_Container >, [3050](#)
- std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, [3134](#)
- std::istreambuf\_iterator< \_CharT, \_Traits >, [3137](#)
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, [3142](#)
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3486](#)
- std::ostreambuf\_iterator< \_CharT, \_Traits >, [3491](#)
- std::pointer\_traits< \_Ptr >, [3523](#)
- std::pointer\_traits< \_Tp \* >, [3524](#)
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, [3550](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3591](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3782](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3815](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3845](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3875](#)
- digits
  - std::\_\_numeric\_limits\_base, [1823](#)
  - std::numeric\_limits< \_Tp >, [3446](#)
- digits10
  - std::\_\_numeric\_limits\_base, [1824](#)
  - std::numeric\_limits< \_Tp >, [3446](#)
- direct\_mask\_range\_hashing\_imp.hpp, [4097](#)
- direct\_mod\_range\_hashing\_imp.hpp, [4097](#)
- discard
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2886](#)
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [3042](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [3152](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [3273](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [3640](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_m, \_\_r >, [3640](#)
- discard\_block\_engine
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2884](#), [2885](#)
- distance
  - SGL, [379](#)
  - std, [698](#)
- do\_compare

- std::collate< \_CharT >, 2731
- std::collate\_byname< \_CharT >, 2737
- do\_curr\_symbol
  - std::moneypunct< \_CharT, \_Intl >, 3306
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3318
- do\_date\_order
  - std::time\_get< \_CharT, \_Inlter >, 3672
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3685
- do\_decimal\_point
  - std::moneypunct< \_CharT, \_Intl >, 3307
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3318
  - std::numpunct< \_CharT >, 3473
  - std::numpunct\_byname< \_CharT >, 3479
- do\_falsename
  - std::numpunct< \_CharT >, 3473
  - std::numpunct\_byname< \_CharT >, 3479
- do\_frac\_digits
  - std::moneypunct< \_CharT, \_Intl >, 3307
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3319
- do\_get
  - std::messages< \_CharT >, 3279
  - std::messages\_byname< \_CharT >, 3283
  - std::money\_get< \_CharT, \_Inlter >, 3292
  - std::num\_get< \_CharT, \_Inlter >, 3408–3415
  - std::time\_get< \_CharT, \_Inlter >, 3672
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3685
- do\_get\_date
  - std::time\_get< \_CharT, \_Inlter >, 3673
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3686
- do\_get\_monthname
  - std::time\_get< \_CharT, \_Inlter >, 3674
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3687
- do\_get\_time
  - std::time\_get< \_CharT, \_Inlter >, 3675
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3688
- do\_get\_weekday
  - std::time\_get< \_CharT, \_Inlter >, 3676
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3689
- do\_get\_year
  - std::time\_get< \_CharT, \_Inlter >, 3676
  - std::time\_get\_byname< \_CharT, \_Inlter >, 3689
- do\_grouping
  - std::moneypunct< \_CharT, \_Intl >, 3308
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3319
  - std::numpunct< \_CharT >, 3473
  - std::numpunct\_byname< \_CharT >, 3480
- do\_hash
  - std::collate< \_CharT >, 2731
  - std::collate\_byname< \_CharT >, 2737
- do\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1682, 1683
  - std::ctype< \_CharT >, 2756, 2757
  - std::ctype< wchar\_t >, 2792, 2793
  - std::ctype\_byname< \_CharT >, 2809, 2810
- do\_narrow
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1683, 1684
  - std::ctype< \_CharT >, 2757, 2758
  - std::ctype< char >, 2775, 2776
  - std::ctype< wchar\_t >, 2793, 2794
  - std::ctype\_byname< \_CharT >, 2810, 2811
  - std::ctype\_byname< char >, 2827, 2829
- do\_neg\_format
  - std::moneypunct< \_CharT, \_Intl >, 3308
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3320
- do\_negative\_sign
  - std::moneypunct< \_CharT, \_Intl >, 3309
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3320
- do\_out
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 1677
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 2692
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2698
  - std::codecvt< char, char, mbstate\_t >, 2702
  - std::codecvt< char16\_t, char, mbstate\_t >, 2707
  - std::codecvt< char32\_t, char, mbstate\_t >, 2712
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2717
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2723
- do\_pos\_format
  - std::moneypunct< \_CharT, \_Intl >, 3309
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3321
- do\_positive\_sign
  - std::moneypunct< \_CharT, \_Intl >, 3310
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3321
- do\_put
  - std::money\_put< \_CharT, \_Outlter >, 3298, 3299
  - std::num\_put< \_CharT, \_Outlter >, 3429–3433
  - std::time\_put< \_CharT, \_Outlter >, 3699
  - std::time\_put\_byname< \_CharT, \_Outlter >, 3703
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1685
  - std::ctype< \_CharT >, 2758
  - std::ctype< wchar\_t >, 2795
  - std::ctype\_byname< \_CharT >, 2811
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1685
  - std::ctype< \_CharT >, 2759
  - std::ctype< wchar\_t >, 2795
  - std::ctype\_byname< \_CharT >, 2812
- do\_thousands\_sep
  - std::moneypunct< \_CharT, \_Intl >, 3310
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3322
  - std::numpunct< \_CharT >, 3474
  - std::numpunct\_byname< \_CharT >, 3480
- do\_tolower
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1686, 1687
  - std::ctype< \_CharT >, 2760



- std::ctype< char >, 2777
- std::ctype< wchar\_t >, 2796
- std::ctype\_byname< \_CharT >, 2813
- std::ctype\_byname< char >, 2829, 2830
- do\_toupper
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1687, 1688
  - std::ctype< \_CharT >, 2761
  - std::ctype< char >, 2778
  - std::ctype< wchar\_t >, 2797
  - std::ctype\_byname< \_CharT >, 2814
  - std::ctype\_byname< char >, 2830, 2831
- do\_transform
  - std::collate< \_CharT >, 2732
  - std::collate\_byname< \_CharT >, 2738
- do\_truncate
  - std::num\_punct< \_CharT >, 3474
  - std::num\_punct\_byname< \_CharT >, 3481
- do\_widen
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1689
  - std::ctype< \_CharT >, 2762, 2763
  - std::ctype< char >, 2779
  - std::ctype< wchar\_t >, 2798, 2799
  - std::ctype\_byname< \_CharT >, 2815, 2816
  - std::ctype\_byname< char >, 2831, 2832
- duration\_cast
  - std::chrono, 828
- Dynamic Bitset., 79
  - operator!=, 80
  - operator<<, 81
  - operator<=, 81
  - operator>, 81
  - operator>>, 82
  - operator>=, 82
  - operator^, 82
  - operator-, 80
  - operator&, 80
  - operator|, 83
- dynamic\_bitset, 4097
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3717–3719
- dynamic\_bitset.tcc, 4098
- dynamic\_pointer\_cast
  - std, 699
- e\_pos
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1600
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 1626
- e\_type
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 1600
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 1625
- eback
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 984
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1048
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1074
  - std::basic\_filebuf< \_CharT, \_Traits >, 1969
  - std::basic\_streambuf< \_CharT, \_Traits >, 2469
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2555
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3939
- ECMAScript
  - std::regex\_constants, 849
- egptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 985
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1048
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1074
  - std::basic\_filebuf< \_CharT, \_Traits >, 1969
  - std::basic\_streambuf< \_CharT, \_Traits >, 2469
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2555
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3939
- egrep
  - std::regex\_constants, 850
- element\_type
  - std::auto\_ptr< \_Tp >, 1949
  - std::pointer\_traits< \_Ptr >, 3523
  - std::pointer\_traits< \_Tp \* >, 3525
- ellint\_1
  - Mathematical Special Functions, 149, 168
- ellint\_1f
  - Mathematical Special Functions, 150
- ellint\_1l
  - Mathematical Special Functions, 150
- ellint\_2
  - Mathematical Special Functions, 150, 168
- ellint\_2f
  - Mathematical Special Functions, 151
- ellint\_2l
  - Mathematical Special Functions, 151
- ellint\_3
  - Mathematical Special Functions, 151, 169
- ellint\_3f
  - Mathematical Special Functions, 152
- ellint\_3l
  - Mathematical Special Functions, 153
- emplace
  - std::deque< \_Tp, \_Alloc >, 2870
  - std::list< \_Tp, \_Alloc >, 3168
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3227
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3338
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3370
  - std::set< \_Key, \_Compare, \_Alloc >, 3600
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3792

- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3824](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3854](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3884](#)
- std::vector< \_Tp, \_Alloc >, [3918](#)
- emplace\_after
  - std::forward\_list< \_Tp, \_Alloc >, [2952](#)
- emplace\_front
  - std::forward\_list< \_Tp, \_Alloc >, [2953](#)
- emplace\_hint
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3229](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3338](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3371](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3600](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3793](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3825](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3855](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3885](#)
- empty
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [916](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1184](#)
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1403](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [1433](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2511](#)
  - std::deque< \_Tp, \_Alloc >, [2871](#)
  - std::experimental::fundamentals\_v1::any, [2912](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2953](#)
  - std::list< \_Tp, \_Alloc >, [3169](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3229](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [3258](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3339](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3371](#)
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3534](#)
  - std::queue< \_Tp, \_Sequence >, [3540](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3601](#)
  - std::stack< \_Tp, \_Sequence >, [3648](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3721](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3794](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3794](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3825](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3855](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3885](#)
  - std::vector< \_Tp, \_Alloc >, [3918](#)
  - enable\_if\_t
    - Metaprogramming, [179](#)
  - enable\_special\_members.h, [4099](#)
  - enc\_filebuf.h, [4099](#)
  - end
    - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [916](#)
    - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, [1097](#)
    - \_\_gnu\_parallel::PseudoSequence< \_Tp, \_DifferenceTp >, [1296](#)
    - \_\_gnu\_pbds::sample\_trie\_access\_traits, [1600](#)
    - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, [1626](#)
  - Numeric Arrays, [252](#), [253](#)
  - std, [699](#), [700](#)
  - std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, [1891](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2054](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2108](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2140](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2205](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2257](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2313](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2357](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2400](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2445](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2512](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2635](#)
  - std::deque< \_Tp, \_Alloc >, [2871](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2953](#), [2954](#)
  - std::ios\_base, [3071](#)
  - std::list< \_Tp, \_Alloc >, [3169](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3230](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [3258](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3339](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3372](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3601](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3794](#), [3795](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3794](#)

- `_Alloc >`, [3826](#), [3827](#)
- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3855](#), [3856](#)
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3886](#), [3887](#)
- `std::vector< _Tp, _Alloc >`, [3918](#), [3919](#)
- `endl`
- `std`, [700](#)
- `ends`
- `std`, [701](#)
- `entry_cmp.hpp`, [4100](#)
- `entry_list_fn_imps.hpp`, [4100](#)
- `entry_metadata_base.hpp`, [4100](#)
- `entry_pred.hpp`, [4101](#)
- `eof`
  - `std::basic_fstream< _CharT, _Traits >`, [2006](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [2072](#)
  - `std::basic_ios< _CharT, _Traits >`, [2125](#)
  - `std::basic_iostream< _CharT, _Traits >`, [2158](#)
  - `std::basic_istream< _CharT, _Traits >`, [2221](#)
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [2276](#)
  - `std::basic_ofstream< _CharT, _Traits >`, [2330](#)
  - `std::basic_ostream< _CharT, _Traits >`, [2373](#)
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, [2418](#)
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [2589](#)
- `eofbit`
  - `std::basic_fstream< _CharT, _Traits >`, [2054](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [2109](#)
  - `std::basic_ios< _CharT, _Traits >`, [2141](#)
  - `std::basic_iostream< _CharT, _Traits >`, [2205](#)
  - `std::basic_istream< _CharT, _Traits >`, [2258](#)
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [2314](#)
  - `std::basic_ofstream< _CharT, _Traits >`, [2358](#)
  - `std::basic_ostream< _CharT, _Traits >`, [2400](#)
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, [2445](#)
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [2636](#)
  - `std::ios_base`, [3071](#)
- `ep_ptr`
  - `__gnu_cxx::enc_filebuf< _CharT >`, [985](#)
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [1048](#)
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, [1075](#)
  - `std::basic_filebuf< _CharT, _Traits >`, [1970](#)
  - `std::basic_streambuf< _CharT, _Traits >`, [2469](#)
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [2556](#)
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3939](#)
- `epsilon`
  - `std::numeric_limits< _Tp >`, [3444](#)
- `eq_by_less.hpp`, [4101](#)
- `equal`
  - Non-Mutating, [217](#), [218](#)
  - `std::istreambuf_iterator< _CharT, _Traits >`, [3140](#)
- `equal_range`
  - Binary Search, [43](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [3230–3232](#)
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3340](#), [3341](#)
  - `std::multiset< _Key, _Compare, _Alloc >`, [3372–3374](#)
  - `std::set< _Key, _Compare, _Alloc >`, [3601–3603](#)
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3795](#), [3796](#)
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3827](#), [3828](#)
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3857](#)
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3887](#), [3888](#)
- `equally_split.h`, [4101](#)
- `equals`
  - `std::tr2::bool_set`, [3711](#)
- `erase`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [917](#), [918](#)
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1184](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [2512](#), [2513](#)
  - `std::deque< _Tp, _Alloc >`, [2871](#), [2872](#)
  - `std::list< _Tp, _Alloc >`, [3169](#), [3170](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [3233](#), [3234](#)
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3342–3344](#)
  - `std::multiset< _Key, _Compare, _Alloc >`, [3374](#), [3375](#)
  - `std::set< _Key, _Compare, _Alloc >`, [3604](#), [3605](#)
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3796–3798](#)
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3828–3830](#)
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3858](#), [3860](#)
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3888–3890](#)
  - `std::vector< _Tp, _Alloc >`, [3919](#), [3920](#)
- `erase_after`
  - `std::forward_list< _Tp, _Alloc >`, [2954](#)
- `erase_can_throw`
  - `__gnu_pbds::container_traits< Cntnr >`, [1362](#)
- `erase_fn_imps.hpp`, [4102–4104](#)
- `erase_if.h`, [4104](#)
- `erase_no_store_hash_fn_imps.hpp`, [4104](#)

- erase\_store\_hash\_fn\_imps.hpp, [4105](#)
- error\_backref
  - std::regex\_constants, [842](#)
- error\_badbrace
  - std::regex\_constants, [842](#)
- error\_badrepeat
  - std::regex\_constants, [842](#)
- error\_brace
  - std::regex\_constants, [842](#)
- error\_brack
  - std::regex\_constants, [842](#)
- error\_collate
  - std::regex\_constants, [843](#)
- error\_complexity
  - std::regex\_constants, [843](#)
- error\_constants.h, [4105](#)
- error\_ctype
  - std::regex\_constants, [843](#)
- error\_escape
  - std::regex\_constants, [843](#)
- error\_paren
  - std::regex\_constants, [843](#)
- error\_range
  - std::regex\_constants, [843](#)
- error\_space
  - std::regex\_constants, [843](#)
- error\_stack
  - std::regex\_constants, [844](#)
- error\_type
  - std::regex\_constants, [841](#)
- event
  - std::basic\_fstream< \_CharT, \_Traits >, [2002](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2068](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2122](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2155](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2219](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2273](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2326](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2370](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2414](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2585](#)
  - std::ios\_base, [3061](#)
- event\_callback
  - std::basic\_fstream< \_CharT, \_Traits >, [2000](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2066](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2118](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2153](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2216](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2271](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2324](#)
- std::basic\_ostream< \_CharT, \_Traits >, [2368](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2412](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2582](#)
- std::ios\_base, [3059](#)
- exception, [4106](#)
- exception.h, [4106](#)
- exception.hpp, [4107](#)
- exception\_defines.h, [4107](#)
- exception\_ptr.h, [4108](#)
- Exceptions, [84](#), [85](#)
  - \_\_verbose\_terminate\_handler, [86](#)
  - current\_exception, [86](#)
  - make\_exception\_ptr, [87](#)
  - rethrow\_exception, [87](#)
  - rethrow\_if\_nested, [87](#)
  - throw\_with\_nested, [87](#)
  - what, [87](#)
- exceptions
  - std::basic\_fstream< \_CharT, \_Traits >, [2006](#), [2007](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2072](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2125](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2158](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2221](#), [2222](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2276](#), [2277](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2330](#), [2331](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2373](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2418](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2589](#)
- exchange
  - std, [701](#)
- exp
  - Complex Numbers, [57](#)
- Experimental, [88](#)
- experimental/algorithm
  - sample, [3981](#)
- experimental/bits/fs\_path.h
  - hash\_value, [4125](#)
  - operator!=, [4126](#)
  - operator<, [4126](#)
  - operator<<, [4126](#)
  - operator<=, [4126](#)
  - operator>, [4127](#)
  - operator>>, [4127](#)
  - operator>=, [4127](#)
  - operator/, [4126](#)
  - operator==, [4127](#)
  - swap, [4128](#)
  - u8path, [4128](#)
- experimental/bits/shared\_ptr.h

- get\_deleter, [4320](#)
- experimental/functional
  - is\_bind\_expression\_v, [4136](#)
  - is\_placeholder\_v, [4136](#)
  - make\_boyer\_moore\_horspool\_searcher, [4135](#)
  - make\_boyer\_moore\_searcher, [4135](#)
  - make\_default\_searcher, [4135](#)
  - not\_fn, [4135](#)
- experimental/iterator
  - make\_ostream\_joiner, [4166](#)
- experimental/numeric
  - gcd, [4222](#)
  - lcm, [4222](#)
- expint
  - Mathematical Special Functions, [153](#), [169](#)
- expintf
  - Mathematical Special Functions, [154](#)
- expintl
  - Mathematical Special Functions, [154](#)
- exponential\_distribution
  - std::exponential\_distribution< \_RealType >, [2927](#)
- extc++.h, [4108](#)
- extended
  - std::regex\_constants, [850](#)
- Extensions, [89](#)
- external\_load\_access
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1347](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1555](#)
- extptr\_allocator.h, [4108](#)
- fabs
  - Complex Numbers, [58](#)
  - std, [701](#)
- facet
  - std::locale::facet, [3199](#)
- fail
  - std::basic\_fstream< \_CharT, \_Traits >, [2007](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2073](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2126](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2159](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2222](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2277](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2331](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2374](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2419](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2590](#)
- failbit
  - std::basic\_fstream< \_CharT, \_Traits >, [2055](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2109](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2141](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2206](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2258](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2314](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2358](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2400](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2446](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2636](#)
  - std::ios\_base, [3071](#)
- failed
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3493](#)
- false\_type
  - Metaprogramming, [179](#)
- falsename
  - std::numpunct< \_CharT >, [3475](#)
  - std::numpunct\_byname< \_CharT >, [3481](#)
- fd
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1049](#)
- features.h, [4109](#)
  - \_GLIBCXX\_BAL\_QUICKSORT, [4109](#)
  - \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS, [4110](#)
  - \_GLIBCXX\_FIND\_EQUAL\_SPLIT, [4110](#)
  - \_GLIBCXX\_FIND\_GROWING\_BLOCKS, [4110](#)
  - \_GLIBCXX\_MERGESORT, [4110](#)
  - \_GLIBCXX\_QUICKSORT, [4111](#)
  - \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING, [4111](#)
  - \_GLIBCXX\_TREE\_FULL\_COPY, [4111](#)
  - \_GLIBCXX\_TREE\_INITIAL\_SPLITTING, [4111](#)
- fenv.h, [4112](#)
- file
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1049](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1075](#)
- filebuf
  - I/O, [114](#)
- filesystem, [4112](#)
- Filesystem TS, [90](#)
  - copy\_options, [94](#)
  - perms, [94](#)
- fill
  - Mutating, [187](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2008](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2073](#), [2074](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2126](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2159](#), [2160](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2223](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2278](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2332](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2374](#), [2375](#)

- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2419, 2420
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2590
- fill\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1308
- fill\_n
  - Mutating, 187
- find
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 918–920
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1185
  - Non-Mutating, 219
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2514, 2515
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3235, 3236
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3344–3346
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3376, 3377
  - std::set< \_Key, \_Compare, \_Alloc >, 3605–3607
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3798, 3800
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3830, 3831
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3862
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3890, 3891
- find.h, 4112
- find\_by\_order
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc >, 1611, 1612
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 1618
- find\_end
  - Non-Mutating, 219, 220
- find\_first
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3722
- find\_first\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 921–923
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1185
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2516–2518
- find\_first\_of
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 923–925
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1186
  - Non-Mutating, 221
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2518–2520
- find\_fn\_imps.hpp, 4113, 4114
- find\_if
  - Non-Mutating, 222
- find\_if\_not
  - Non-Mutating, 223
- find\_increasing\_factor
  - \_\_gnu\_parallel::Settings, 1308
- find\_initial\_block\_size
  - \_\_gnu\_parallel::Settings, 1308
- find\_last\_not\_of
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 925–927
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1186
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2520–2522
- find\_last\_of
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 928, 929
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1187
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2522–2524
- find\_maximum\_block\_size
  - \_\_gnu\_parallel::Settings, 1308
- find\_next
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3722
- find\_no\_store\_hash\_fn\_imps.hpp, 4114
- find\_scale\_factor
  - \_\_gnu\_parallel::Settings, 1309
- find\_selectors.h, 4115
- find\_sequential\_search\_size
  - \_\_gnu\_parallel::Settings, 1309
- find\_store\_hash\_fn\_imps.hpp, 4115
- first
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1258
  - std::pair< \_T1, \_T2 >, 3505
  - std::sub\_match< \_Biliter >, 3659
- first\_argument\_type
  - \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, 1023
  - \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, 1024
  - \_\_gnu\_parallel::EqualFromLess< \_T1, \_T2, \_Compare >, 1250
  - \_\_gnu\_parallel::EqualTo< \_T1, \_T2 >, 1251
  - \_\_gnu\_parallel::Less< \_T1, \_T2 >, 1261
  - \_\_gnu\_parallel::Lexicographic< \_T1, \_T2, \_Compare >, 1263
  - \_\_gnu\_parallel::LexicographicReverse< \_T1, \_T2, \_Compare >, 1265
  - \_\_gnu\_parallel::Multiplies< \_Tp1, \_Tp2, \_Result >, 1288



- `__gnu_parallel::Plus< _Tp1, _Tp2, Result >`, 1292
- `std::Maybe_unary_or_binary_function< _Res, _T1, _T2 >`, 1884
- `std::binary_function< _Arg1, _Arg2, Result >`, 2646
- `std::binary_negate< _Predicate >`, 2648
- `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 2748
- `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 2750
- `std::divides< _Tp >`, 2895
- `std::equal_to< _Tp >`, 2900
- `std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >`, 2923
- `std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >`, 2924
- `std::greater< _Tp >`, 3004
- `std::greater_equal< _Tp >`, 3007
- `std::less< _Tp >`, 3146
- `std::less_equal< _Tp >`, 3149
- `std::logical_and< _Tp >`, 3205
- `std::logical_or< _Tp >`, 3209
- `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 3265
- `std::mem_fun1_t< _Ret, _Tp, _Arg >`, 3267
- `std::minus< _Tp >`, 3284
- `std::modulus< _Tp >`, 3287
- `std::multiplies< _Tp >`, 3359
- `std::not_equal_to< _Tp >`, 3402
- `std::owner_less< shared_ptr< _Tp > >`, 3498
- `std::owner_less< void >`, 3499
- `std::owner_less< weak_ptr< _Tp > >`, 3500
- `std::plus< _Tp >`, 3518
- `std::pointer_to_binary_function< _Arg1, _Arg2, Result >`, 3520
- fixed**
  - `std`, 701
  - `std::basic_fstream< _CharT, _Traits >`, 2055
  - `std::basic_ifstream< _CharT, _Traits >`, 2109
  - `std::basic_ios< _CharT, _Traits >`, 2141
  - `std::basic_iostream< _CharT, _Traits >`, 2206
  - `std::basic_istream< _CharT, _Traits >`, 2258
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2314
  - `std::basic_ofstream< _CharT, _Traits >`, 2358
  - `std::basic_ostream< _CharT, _Traits >`, 2401
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 2446
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2636
  - `std::ios_base`, 3072
- flags**
  - `std::basic_fstream< _CharT, _Traits >`, 2009
  - `std::basic_ifstream< _CharT, _Traits >`, 2074, 2075
  - `std::basic_ios< _CharT, _Traits >`, 2127
  - `std::basic_iostream< _CharT, _Traits >`, 2160, 2161
  - `std::basic_istream< _CharT, _Traits >`, 2224
- `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2279
- `std::basic_ofstream< _CharT, _Traits >`, 2333
- `std::basic_ostream< _CharT, _Traits >`, 2375, 2376
- `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 2420, 2421
- `std::basic_regex< _Ch_type, _Rx_traits >`, 2459
- `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2591
- `std::ios_base`, 3062
- flip**
  - `std::bitset< _Nb >`, 2664, 2665
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 3723
- float\_denorm\_style**
  - `std`, 672
- float\_round\_style**
  - `std`, 673
- floatfield**
  - `std::basic_fstream< _CharT, _Traits >`, 2055
  - `std::basic_ifstream< _CharT, _Traits >`, 2109
  - `std::basic_ios< _CharT, _Traits >`, 2141
  - `std::basic_iostream< _CharT, _Traits >`, 2206
  - `std::basic_istream< _CharT, _Traits >`, 2258
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2314
  - `std::basic_ofstream< _CharT, _Traits >`, 2358
  - `std::basic_ostream< _CharT, _Traits >`, 2401
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 2446
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2636
  - `std::ios_base`, 3072
- flush**
  - `std`, 702
  - `std::basic_fstream< _CharT, _Traits >`, 2009
  - `std::basic_iostream< _CharT, _Traits >`, 2161
  - `std::basic_ofstream< _CharT, _Traits >`, 2333
  - `std::basic_ostream< _CharT, _Traits >`, 2376
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 2421
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2592
- fmtflags**
  - `std::basic_fstream< _CharT, _Traits >`, 2000
  - `std::basic_ifstream< _CharT, _Traits >`, 2066
  - `std::basic_ios< _CharT, _Traits >`, 2119
  - `std::basic_iostream< _CharT, _Traits >`, 2153
  - `std::basic_istream< _CharT, _Traits >`, 2217
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2271
  - `std::basic_ofstream< _CharT, _Traits >`, 2324
  - `std::basic_ostream< _CharT, _Traits >`, 2368
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 2412

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2583
- std::ios\_base, 3059
- for\_each
  - Non-Mutating, 223
- for\_each.h, 4115
- for\_each\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1309
- for\_each\_selectors.h, 4116
- format
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3258, 3259
- format\_default
  - std::regex\_constants, 850
- format\_first\_only
  - std::regex\_constants, 851
- format\_no\_copy
  - std::regex\_constants, 851
- format\_sed
  - std::regex\_constants, 851
- formatter.h, 4116
- forward
  - Utilities, 438
- forward\_list, 4117–4119
  - std::forward\_list< \_Tp, \_Alloc >, 2945–2948
- forward\_list.h, 4120
- forward\_list.tcc, 4121
- fpos
  - std::fpos< \_StateT >, 2968
- frac\_digits
  - std::moneypunct< \_CharT, \_Intl >, 3311
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3322
- from\_bytes
  - std::wstring\_convert< \_Codecv, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3964, 3965
- front
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 930
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1187, 1188
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2524, 2525
  - std::deque< \_Tp, \_Alloc >, 2872
  - std::forward\_list< \_Tp, \_Alloc >, 2955
  - std::list< \_Tp, \_Alloc >, 3170, 3171
  - std::queue< \_Tp, \_Sequence >, 3540
  - std::vector< \_Tp, \_Alloc >, 3920
- front\_insert\_iterator
  - std::front\_insert\_iterator< \_Container >, 2973
- front\_inserter
  - Iterators, 126
- fs\_dir.h, 4122
- fs\_fwd.h, 4122, 4123
- fs\_path.h, 4124, 4125
- fstream, 4129
  - I/O, 114
- fstream.tcc, 4129
- functexcept.h, 4130
- function
  - std::function< \_Res(\_ArgTypes...) >, 2976, 2977
- Function Objects, 95
  - mem\_fn, 96
- functional, 4131, 4133, 4134
- functional\_hash.h, 4136
- functions.h, 4137
- future, 4139
  - std::future< \_Res >, 2985
  - std::future< \_Res & >, 2988
  - std::future< void >, 2991
- future\_category
  - Futures, 99
- future\_errc
  - Futures, 98
- future\_status
  - Futures, 98
- Futures, 97
  - async, 99
  - future\_category, 99
  - future\_errc, 98
  - future\_status, 98
  - launch, 99
  - make\_error\_code, 100
  - make\_error\_condition, 100
  - swap, 100
- gamma\_distribution
  - std::gamma\_distribution< \_RealType >, 2994
- gump
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 985
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1049
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1075
  - std::basic\_filebuf< \_CharT, \_Traits >, 1970
  - std::basic\_streambuf< \_CharT, \_Traits >, 2470
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2556
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, 3940
- gcd
  - experimental/numeric, 4222
- gcount
  - std::basic\_fstream< \_CharT, \_Traits >, 2010
  - std::basic\_ifstream< \_CharT, \_Traits >, 2075
  - std::basic\_iostream< \_CharT, \_Traits >, 2161
  - std::basic\_istream< \_CharT, \_Traits >, 2224
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2279
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2592
- generate
  - Mutating, 189



- generate\_canonical
  - Random Number Generation, [316](#)
- generate\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1309](#)
- generate\_n
  - Mutating, [189](#)
- get
  - \_\_gnu\_parallel::Settings, [1306](#)
  - std::\_\_allocated\_ptr<\_Alloc>, [1669](#)
  - std::auto\_ptr<\_Tp>, [1951](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [2010–2013](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2075–2078](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2162–2165](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2225–2228](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2280–2283](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2592–2595](#)
  - std::future<\_Res>, [2985](#)
  - std::future<\_Res &>, [2988](#)
  - std::future<void>, [2991](#)
  - std::money\_get<\_CharT, \_InIter>, [3293, 3294](#)
  - std::num\_get<\_CharT, \_InIter>, [3415–3424](#)
  - std::shared\_future<\_Res>, [3619](#)
  - std::shared\_future<\_Res &>, [3623](#)
  - std::time\_get<\_CharT, \_InIter>, [3677, 3678](#)
  - std::time\_get\_byname<\_CharT, \_InIter>, [3690, 3691](#)
  - std::unique\_ptr<\_Tp, \_Dp>, [3767](#)
  - std::unique\_ptr<\_Tp[], \_Dp>, [3774](#)
  - Utilities, [439, 440](#)
- get\_actual\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy<Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type>, [1559](#)
- get\_allocator
  - \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>, [930](#)
  - \_\_gnu\_debug::basic\_string<\_CharT, \_Traits, \_Allocator>, [1188](#)
  - std::basic\_string<\_CharT, \_Traits, \_Alloc>, [2525](#)
  - std::deque<\_Tp, \_Alloc>, [2873](#)
  - std::forward\_list<\_Tp, \_Alloc>, [2955](#)
  - std::list<\_Tp, \_Alloc>, [3171](#)
  - std::map<\_Key, \_Tp, \_Compare, \_Alloc>, [3237](#)
  - std::match\_results<\_Bi\_iter, \_Alloc>, [3260](#)
  - std::multimap<\_Key, \_Tp, \_Compare, \_Alloc>, [3346](#)
  - std::multiset<\_Key, \_Compare, \_Alloc>, [3378](#)
  - std::set<\_Key, \_Compare, \_Alloc>, [3607](#)
  - std::tr2::dynamic\_bitset<\_WordT, \_Alloc>, [3723](#)
  - std::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc>, [3800](#)
  - std::unordered\_multimap<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc>, [3831](#)
  - std::unordered\_multiset<\_Value, \_Hash, \_Pred, \_Alloc>, [3863](#)
  - std::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc>, [3891](#)
  - std::vector<\_Tp, \_Alloc>, [3921](#)
- get\_child
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer<Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc>, [1484](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter<Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc>, [1488](#)
- get\_comb\_hash\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map<Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy>, [1404](#)
- get\_comb\_probe\_fn
  - \_\_gnu\_pbds::detail::gp\_ht\_map<Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy>, [1433](#)
- get\_date
  - std::time\_get<\_CharT, \_InIter>, [3679](#)
  - std::time\_get\_byname<\_CharT, \_InIter>, [3692](#)
- get\_deleter
  - experimental/bits/shared\_ptr.h, [4320](#)
  - Pointer Abstractions, [305](#)
  - std::unique\_ptr<\_Tp, \_Dp>, [3767](#)
  - std::unique\_ptr<\_Tp[], \_Dp>, [3774](#)
- get\_eq\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map<Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy>, [1404](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map<Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy>, [1434](#)
- get\_hash\_fn
  - \_\_gnu\_pbds::detail::cc\_ht\_map<Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy>, [1405](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map<Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy>, [1434](#)
- get\_id
  - std::this\_thread, [858](#)
- get\_l\_child
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it<Node, Const\_Iterator, Iterator, \_Alloc>, [1372](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it<Node, Const\_Iterator, Iterator, \_Alloc>, [1378](#)

- \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [1463](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [1464](#)
- get\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1348](#)
- get\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1555](#)
- get\_metadata
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1372](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1379](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [1484](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [1488](#)
- get\_money
  - std, [702](#)
- get\_monthname
  - std::time\_get< \_CharT, \_InIter >, [3679](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3692](#)
- get\_nearest\_larger\_size
  - \_\_gnu\_pbds::sample\_size\_policy, [1598](#)
- get\_nearest\_smaller\_size
  - \_\_gnu\_pbds::sample\_size\_policy, [1598](#)
- get\_new\_handler
  - std, [702](#)
- get\_new\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [1560](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1590](#)
- get\_probe\_fn
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [1435](#)
- get\_r\_child
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1373](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1379](#)
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [1463](#)
  - \_\_gnu\_pbds::detail::ov\_tree\_node\_it\_< Value\_Type, Metadata\_Type, \_Alloc >, [1465](#)
- get\_resize\_policy
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [1405](#)
- \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [1435](#)
- get\_size\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [1560](#)
- get\_temporary\_buffer
  - std, [702](#)
- get\_terminate
  - std, [703](#)
- get\_time
  - std, [703](#)
  - std::time\_get< \_CharT, \_InIter >, [3680](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3693](#)
- get\_trigger\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [1560](#), [1561](#)
- get\_unexpected
  - std, [704](#)
- get\_weekday
  - std::time\_get< \_CharT, \_InIter >, [3681](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3694](#)
- get\_year
  - std::time\_get< \_CharT, \_InIter >, [3682](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3695](#)
- getline
  - std, [704–706](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2013–2015](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2079](#), [2080](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2165](#), [2166](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2228](#), [2229](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2283](#), [2284](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2596](#), [2597](#)
- getloc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [986](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1050](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1076](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1971](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2015](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2080](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2128](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2167](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2230](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2285](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2334](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2376](#)

- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2421
- std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 2460
- std::basic\_streambuf< \_CharT, \_Traits >, 2470
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2557
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2597
- std::ios\_base, 3063
- std::regex\_traits< \_Ch\_type >, 3566
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3940
- global
  - std::locale, 3191
- good
  - std::basic\_fstream< \_CharT, \_Traits >, 2015
  - std::basic\_ifstream< \_CharT, \_Traits >, 2080
  - std::basic\_ios< \_CharT, \_Traits >, 2128
  - std::basic\_iostream< \_CharT, \_Traits >, 2167
  - std::basic\_istream< \_CharT, \_Traits >, 2230
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2285
  - std::basic\_ofstream< \_CharT, \_Traits >, 2334
  - std::basic\_ostream< \_CharT, \_Traits >, 2377
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2422
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2597
- goodbit
  - std::basic\_fstream< \_CharT, \_Traits >, 2055
  - std::basic\_ifstream< \_CharT, \_Traits >, 2110
  - std::basic\_ios< \_CharT, \_Traits >, 2142
  - std::basic\_iostream< \_CharT, \_Traits >, 2206
  - std::basic\_istream< \_CharT, \_Traits >, 2259
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2315
  - std::basic\_ofstream< \_CharT, \_Traits >, 2359
  - std::basic\_ostream< \_CharT, \_Traits >, 2401
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2446
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2637
  - std::ios\_base, 3072
- gp\_hash\_table
  - \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, 1546–1550
- gp\_ht\_map.hpp, 4141
- gptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 986
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1050
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1076
  - std::basic\_filebuf< \_CharT, \_Traits >, 1971
  - std::basic\_streambuf< \_CharT, \_Traits >, 2470
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2557
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3940
- grep
  - std::regex\_constants, 851
- grouping
  - std::moneypunct< \_CharT, \_Intl >, 3311
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3323
  - std::numpunct< \_CharT >, 3475
  - std::numpunct\_byname< \_CharT >, 3481
- gslice
  - Numeric Arrays, 245, 246
- gslice.h, 4141
- gslice\_array
  - Numeric Arrays, 246
- gslice\_array.h, 4142
- has\_denorm
  - std::\_\_numeric\_limits\_base, 1824
  - std::numeric\_limits< \_Tp >, 3446
- has\_denorm\_loss
  - std::\_\_numeric\_limits\_base, 1824
  - std::numeric\_limits< \_Tp >, 3446
- has\_facet
  - Locales, 131
  - std::locale, 3194
  - std::locale::id, 3201
- has\_infinity
  - std::\_\_numeric\_limits\_base, 1824
  - std::numeric\_limits< \_Tp >, 3446
- has\_quiet\_NaN
  - std::\_\_numeric\_limits\_base, 1824
  - std::numeric\_limits< \_Tp >, 3447
- has\_signaling\_NaN
  - std::\_\_numeric\_limits\_base, 1825
  - std::numeric\_limits< \_Tp >, 3447
- hash
  - std::collate< \_CharT >, 2733
  - std::collate\_byname< \_CharT >, 2739
- Hash-Based, 101
- hash\_bytes.h, 4142
- hash\_eq\_fn.hpp, 4142
- hash\_exponential\_size\_policy
  - \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type >, 1553
- hash\_exponential\_size\_policy\_imp.hpp, 4143
- hash\_fun.h, 4143
- hash\_function
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3800
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3831
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3863
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3891

- hash\_load\_check\_resize\_trigger
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< Ex-  
ternal\_Load\_Access, Size\_Type >, 1555
- hash\_load\_check\_resize\_trigger\_imp.hpp, 4143
- hash\_load\_check\_resize\_trigger\_size\_base.hpp, 4144
- hash\_map, 4144
- hash\_policy.hpp, 4145
- hash\_prime\_size\_policy
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, 1557
- hash\_prime\_size\_policy\_imp.hpp, 4146
- hash\_set, 4146
- hash\_standard\_resize\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Pol-  
icy, Trigger\_Policy, External\_Size\_Access,  
Size\_Type >, 1559
- hash\_standard\_resize\_policy\_imp.hpp, 4147
- hash\_value
  - experimental/bits/fs\_path.h, 4125
- hasher
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
loc >, 3782
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
\_Alloc >, 3815
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
loc >, 3845
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
>, 3875
- Hashes, 102
- hashtable.h, 4147, 4148
- hashtable\_policy.h, 4148
- Heap, 103
  - is\_heap, 103, 104
  - is\_heap\_until, 104, 105
  - make\_heap, 105, 106
  - pop\_heap, 106, 107
  - push\_heap, 107, 108
  - sort\_heap, 108, 109
- Heap-Based, 110
  - priority\_queue, 111
- helper\_functions.h, 4150
- hermite
  - Mathematical Special Functions, 154, 169
- hermitef
  - Mathematical Special Functions, 155
- hermitel
  - Mathematical Special Functions, 155
- hex
  - std, 707
  - std::basic\_fstream< \_CharT, \_Traits >, 2056
  - std::basic\_ifstream< \_CharT, \_Traits >, 2110
  - std::basic\_ios< \_CharT, \_Traits >, 2142
  - std::basic\_iostream< \_CharT, \_Traits >, 2207
  - std::basic\_istream< \_CharT, \_Traits >, 2259
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 2315
  - std::basic\_ofstream< \_CharT, \_Traits >, 2359
  - std::basic\_ostream< \_CharT, \_Traits >, 2401
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2447
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2637
  - std::ios\_base, 3072
- hexfloat
  - std, 707
- high\_resolution\_clock
  - chrono, 4042
- hours
  - std::chrono, 827
- hyperg
  - \_\_gnu\_cxx, 465
  - std::tr1, 862
- hypergf
  - \_\_gnu\_cxx, 466
- hypergl
  - \_\_gnu\_cxx, 466
- I/O, 113
  - filebuf, 114
  - fstream, 114
  - ifstream, 115
  - ios, 115
  - iostream, 115
  - istream, 115
  - istringstream, 115
  - ofstream, 116
  - ostream, 116
  - ostringstream, 116
  - streambuf, 116
  - stringbuf, 116
  - stringstream, 117
  - wfilebuf, 117
  - wfstream, 117
  - wifstream, 117
  - wios, 117
  - wiostream, 118
  - wistream, 118
  - wistringstream, 118
  - wofstream, 118
  - wostream, 118
  - wostringstream, 119
  - wstreambuf, 119
  - wstringbuf, 119
  - wstringstream, 119
- icase
  - std::regex\_constants, 852
- id
  - std::collate< \_CharT >, 2734

- std::collate\_byname< \_CharT >, 2740
- std::ctype< \_CharT >, 2771
- std::ctype< char >, 2788
- std::ctype< wchar\_t >, 2806
- std::ctype\_byname< \_CharT >, 2824
- std::ctype\_byname< char >, 2841
- std::locale::id, 3200
- std::messages< \_CharT >, 3280
- std::messages\_byname< \_CharT >, 3283
- std::money\_get< \_CharT, \_InIter >, 3295
- std::money\_put< \_CharT, \_OutIter >, 3301
- std::moneypunct< \_CharT, \_Intl >, 3314
- std::moneypunct\_byname< \_CharT, \_Intl >, 3326
- std::num\_get< \_CharT, \_InIter >, 3425
- std::num\_put< \_CharT, \_OutIter >, 3442
- std::numpunct< \_CharT >, 3477
- std::numpunct\_byname< \_CharT >, 3483
- std::time\_get< \_CharT, \_InIter >, 3682
- std::time\_get\_byname< \_CharT, \_InIter >, 3695
- std::time\_put< \_CharT, \_OutIter >, 3701
- std::time\_put\_byname< \_CharT, \_OutIter >, 3705
- identity\_element
  - SGL, 379, 380
- ifstream
  - I/O, 115
- ignore
  - std::basic\_fstream< \_CharT, \_Traits >, 2015, 2016
  - std::basic\_ifstream< \_CharT, \_Traits >, 2081, 2082
  - std::basic\_iostream< \_CharT, \_Traits >, 2167, 2168
  - std::basic\_istream< \_CharT, \_Traits >, 2230, 2231
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2285, 2286
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2598, 2599
- imbue
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 986
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1050
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1076
  - std::basic\_filebuf< \_CharT, \_Traits >, 1971
  - std::basic\_fstream< \_CharT, \_Traits >, 2017
  - std::basic\_ifstream< \_CharT, \_Traits >, 2082
  - std::basic\_ios< \_CharT, \_Traits >, 2128
  - std::basic\_iostream< \_CharT, \_Traits >, 2169
  - std::basic\_istream< \_CharT, \_Traits >, 2232
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2287
  - std::basic\_ofstream< \_CharT, \_Traits >, 2334
  - std::basic\_ostream< \_CharT, \_Traits >, 2377
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 2422
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 2460
  - std::basic\_streambuf< \_CharT, \_Traits >, 2471
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2557
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2599
  - std::ios\_base, 3063
  - std::regex\_traits< \_Ch\_type >, 3566
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3941
- in
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 1677
  - std::basic\_fstream< \_CharT, \_Traits >, 2056
  - std::basic\_ifstream< \_CharT, \_Traits >, 2110
  - std::basic\_ios< \_CharT, \_Traits >, 2142
  - std::basic\_iostream< \_CharT, \_Traits >, 2207
  - std::basic\_istream< \_CharT, \_Traits >, 2259
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2315
  - std::basic\_ofstream< \_CharT, \_Traits >, 2359
  - std::basic\_ostream< \_CharT, \_Traits >, 2402
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 2447
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2637
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 2693
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2698
  - std::codecvt< char, char, mbstate\_t >, 2703
  - std::codecvt< char16\_t, char, mbstate\_t >, 2708
  - std::codecvt< char32\_t, char, mbstate\_t >, 2713
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2718
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2724
  - std::ios\_base, 3073
- in\_avail
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 987
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1051
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1077
  - std::basic\_filebuf< \_CharT, \_Traits >, 1972
  - std::basic\_streambuf< \_CharT, \_Traits >, 2471
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2558
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3941
- in\_place
  - Optional values, 287
- includes
  - Set Operation, 386
- increment
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 3156
- independent\_bits\_engine
  - std::independent\_bits\_engine< \_\_RandomNumberEngine, \_\_w, \_UIntType >, 3040–3042
- index\_sequence
  - std, 670
- index\_sequence\_for
  - std, 670

- indirect\_array
  - Numeric Arrays, [246](#)
- indirect\_array.h, [4151](#)
- infinity
  - std::numeric\_limits< \_Tp >, [3444](#)
- info\_fn\_imps.hpp, [4152](#), [4153](#)
- init
  - std::basic\_fstream< \_CharT, \_Traits >, [2017](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2083](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2129](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2169](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2232](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2287](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2335](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2378](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2423](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2600](#)
- initializer\_list, [4153](#)
- inner\_product
  - std, [707](#), [708](#)
- inplace\_merge
  - Sorting, [396](#)
- insert
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [931–936](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1188–1191](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2525–2530](#)
  - std::deque< \_Tp, \_Alloc >, [2873–2875](#)
  - std::list< \_Tp, \_Alloc >, [3171–3173](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3237–3241](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3346–3351](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3378–3380](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3607–3609](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3801–3805](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3832–3836](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3863–3866](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3892–3895](#)
  - std::vector< \_Tp, \_Alloc >, [3921–3923](#)
- insert\_after
  - std::forward\_list< \_Tp, \_Alloc >, [2956](#), [2957](#)
- insert\_fn\_imps.hpp, [4154–4156](#)
- insert\_iterator
  - std::insert\_iterator< \_Container >, [3051](#)
- insert\_join\_fn\_imps.hpp, [4156](#)
- insert\_no\_store\_hash\_fn\_imps.hpp, [4156](#)
- insert\_store\_hash\_fn\_imps.hpp, [4156](#)
- inserter
  - Iterators, [127](#)
- int\_type
  - std::basic\_ios< \_CharT, \_Traits >, [2119](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2467](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [3138](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3937](#)
- internal
  - std, [708](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2056](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2110](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2142](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2207](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2259](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2315](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2359](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2402](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2447](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2637](#)
  - std::ios\_base, [3073](#)
- intervals
  - std::piecewise\_constant\_distribution< \_RealType >, [3507](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [3513](#)
- intl
  - std::moneypunct< \_CharT, \_Intl >, [3315](#)
- Invalidation Guarantees, [120](#)
- invoke.h, [4156](#)
- io\_errc
  - std, [673](#)
- iomanip, [4157](#)
- ios, [4159](#)
  - I/O, [115](#)
- ios\_base.h, [4159](#)
- iosfwd, [4161](#)
- iostate
  - std::basic\_fstream< \_CharT, \_Traits >, [2001](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2067](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2120](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2154](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2217](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2272](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2325](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2369](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2413](#)



- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2583
- std::ios\_base, 3060
- iostream, 4162
  - I/O, 115
- iota
  - std, 709
- is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 1690
  - std::ctype< \_CharT >, 2763, 2764
  - std::ctype< char >, 2780
  - std::ctype< wchar\_t >, 2799, 2800
  - std::ctype\_byname< \_CharT >, 2816, 2817
  - std::ctype\_byname< char >, 2833
- is\_always\_equal
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 867
  - std::allocator\_traits< \_Alloc >, 1910
  - std::allocator\_traits< allocator< \_Tp > >, 1918
- is\_bind\_expression\_v
  - experimental/functional, 4136
- is\_bounded
  - std::\_\_numeric\_limits\_base, 1825
  - std::numeric\_limits< \_Tp >, 3447
- is\_emptyset
  - std::tr2::bool\_set, 3711
- is\_exact
  - std::\_\_numeric\_limits\_base, 1825
  - std::numeric\_limits< \_Tp >, 3447
- is\_grow\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1348
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1594
- is\_heap
  - Heap, 103, 104
- is\_heap\_until
  - Heap, 104, 105
- is\_iec559
  - std::\_\_numeric\_limits\_base, 1825
  - std::numeric\_limits< \_Tp >, 3447
- is\_indeterminate
  - std::tr2::bool\_set, 3712
- is\_integer
  - std::\_\_numeric\_limits\_base, 1825
  - std::numeric\_limits< \_Tp >, 3448
- is\_modulo
  - std::\_\_numeric\_limits\_base, 1826
  - std::numeric\_limits< \_Tp >, 3448
- is\_nothrow\_swappable\_v
  - std, 790
- is\_nothrow\_swappable\_with\_v
  - std, 790
- is\_open
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 987
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1051
- std::basic\_filebuf< \_CharT, \_Traits >, 1972
- std::basic\_fstream< \_CharT, \_Traits >, 2018
- std::basic\_ifstream< \_CharT, \_Traits >, 2083
- std::basic\_ofstream< \_CharT, \_Traits >, 2335
- is\_partitioned
  - Mutating, 190
- is\_permutation
  - Non-Mutating, 224, 226
- is\_placeholder\_v
  - experimental/functional, 4136
- is\_resize\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1348
  - \_\_gnu\_pbds::sample\_resize\_policy, 1590
  - \_\_gnu\_pbds::sample\_resize\_trigger, 1594
- is\_signed
  - std::\_\_numeric\_limits\_base, 1826
  - std::numeric\_limits< \_Tp >, 3448
- is\_singleton
  - std::tr2::bool\_set, 3712
- is\_sorted
  - Sorting, 397, 398
- is\_sorted\_until
  - Sorting, 398, 399
- is\_specialized
  - std::\_\_numeric\_limits\_base, 1826
  - std::numeric\_limits< \_Tp >, 3448
- is\_swappable\_v
  - std, 790
- is\_swappable\_with\_v
  - std, 791
- isalnum
  - std, 709
- isalpha
  - std, 709
- isblank
  - std, 710
- iscntrl
  - std, 710
- isctype
  - std::regex\_traits< \_Ch\_type >, 3566
- isdigit
  - std, 710
- isgraph
  - std, 710
- islower
  - std, 711
- isprint
  - std, 711
- ispunct
  - std, 711
- isspace
  - std, 711
- istream, 4163

- I/O, [115](#)
- istream.tcc, [4164](#)
- istream\_iterator
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, [3135](#)
- istream\_type
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [3138](#)
- istreambuf\_iterator
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [3139](#), [3140](#)
- istreamstring
  - I/O, [115](#)
- isupper
  - std, [712](#)
- isxdigit
  - std, [712](#)
- iter\_swap
  - Mutating, [191](#)
- iter\_type
  - std::money\_get< \_CharT, \_InIter >, [3291](#)
  - std::money\_put< \_CharT, \_OutIter >, [3297](#)
  - std::num\_get< \_CharT, \_InIter >, [3407](#)
  - std::num\_put< \_CharT, \_OutIter >, [3427](#)
  - std::time\_get< \_CharT, \_InIter >, [3671](#)
  - std::time\_put< \_CharT, \_OutIter >, [3698](#)
- iterator, [4165](#), [4166](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3591](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3783](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3816](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3845](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3875](#)
- Iterator Tags, [121](#)
- iterator.h, [4167](#)
- iterator.hpp, [4167](#)
- iterator\_, [1652](#)
  - const\_pointer, [1654](#)
  - const\_reference, [1654](#)
  - difference\_type, [1654](#)
  - iterator\_, [1655](#)
  - iterator\_category, [1654](#)
  - m\_p\_tbl, [1658](#)
  - operator const point\_iterator\_, [1655](#)
  - operator point\_iterator\_, [1656](#)
  - operator!=, [1656](#)
  - operator\*, [1656](#)
  - operator++, [1656](#), [1657](#)
  - operator->, [1657](#)
  - operator==, [1657](#)
  - pointer, [1654](#)
  - reference, [1654](#)
  - value\_type, [1655](#)
- iterator\_category
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1371](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, [1377](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, [1386](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, [1391](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, [1442](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, [1448](#)
- const\_iterator\_, [1649](#)
- iterator\_, [1654](#)
- point\_const\_iterator\_, [1660](#)
- point\_iterator\_, [1664](#)
- std::back\_insert\_iterator< \_Container >, [1955](#)
- std::front\_insert\_iterator< \_Container >, [2972](#)
- std::insert\_iterator< \_Container >, [3050](#)
- std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, [3134](#)
- std::istreambuf\_iterator< \_CharT, \_Traits >, [3138](#)
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, [3142](#)
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3487](#)
- std::ostreambuf\_iterator< \_CharT, \_Traits >, [3491](#)
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, [3550](#)
- std::reverse\_iterator< \_Iterator >, [3577](#)
- iterator\_fn\_imps.hpp, [4167](#)
- iterator\_tracker.h, [4167](#)
- Iterators, [122](#)
  - \_\_iterator\_category, [125](#)
  - back\_inserter, [126](#)
  - front\_inserter, [126](#)
  - inserter, [127](#)
  - make\_reverse\_iterator, [127](#)
  - operator!=, [127](#)
  - operator==, [128](#)
- iterators\_fn\_imps.hpp, [4169](#), [4170](#)
- isword
  - std::basic\_fstream< \_CharT, \_Traits >, [2018](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2083](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2129](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2170](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2233](#)
  - std::basic\_istreamstring< \_CharT, \_Traits, \_Alloc >, [2288](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2335](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2378](#)
  - std::basic\_ostreamstring< \_CharT, \_Traits, \_Alloc >, [2423](#)



- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2600
- std::ios\_base, 3064
- k
  - std::negative\_binomial\_distribution< \_IntType >, 3391
- key\_comp
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3242
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3351
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3380
  - std::set< \_Key, \_Compare, \_Alloc >, 3610
- key\_compare
  - std::set< \_Key, \_Compare, \_Alloc >, 3591
- key\_eq
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3805
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3836
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3866
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3895
- key\_equal
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3783
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3816
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3846
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3876
- key\_type
  - std::set< \_Key, \_Compare, \_Alloc >, 3592
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3783
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3816
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3846
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3876
- kill\_dependency
  - Atomics, 28
- L1\_cache\_size
  - \_\_gnu\_parallel::Settings, 1309
- L2\_cache\_size
  - \_\_gnu\_parallel::Settings, 1310
- laguerre
  - Mathematical Special Functions, 155, 169
- laguerref
  - Mathematical Special Functions, 156
- laguerrel
  - Mathematical Special Functions, 156
- lambda
  - std::exponential\_distribution< \_RealType >, 2928
- launch
  - Futures, 99
- lcm
  - experimental/numeric, 4222
- left
  - std, 712
  - std::basic\_fstream< \_CharT, \_Traits >, 2056
  - std::basic\_ifstream< \_CharT, \_Traits >, 2111
  - std::basic\_ios< \_CharT, \_Traits >, 2143
  - std::basic\_iostream< \_CharT, \_Traits >, 2207
  - std::basic\_istream< \_CharT, \_Traits >, 2260
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2316
  - std::basic\_ofstream< \_CharT, \_Traits >, 2360
  - std::basic\_ostream< \_CharT, \_Traits >, 2402
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2447
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2638
  - std::ios\_base, 3073
  - left\_child\_next\_sibling\_heap.hpp, 4170
  - left\_child\_next\_sibling\_heap\_const\_iterator\_
    - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, 1443, 1444
  - left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
    - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, 1449, 1450
- legendre
  - Mathematical Special Functions, 156, 170
- legendref
  - Mathematical Special Functions, 157
- legendrel
  - Mathematical Special Functions, 157
- length
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 937
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1192
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2531
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3260
  - std::regex\_traits< \_Ch\_type >, 3567
  - std::sub\_match< \_Biter >, 3658
- lexicographical\_compare
  - Sorting, 399, 400
- lexicographical\_compare\_3way
  - SGI, 380
- lfts\_config.h, 4170
- limits, 4171
- linear\_congruential\_engine
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 3151, 3152

- linear\_probe\_fn\_imp.hpp, 4172
- list, 4172–4174
  - std::list< \_Tp, \_Alloc >, 3161–3163
- List-Based, 129
- list.tcc, 4175
- list\_partition
  - \_\_gnu\_parallel, 541
- list\_partition.h, 4175
- list\_update
  - \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, 1566
- list\_update\_policy.hpp, 4175
- load\_factor
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3806
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3836
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3867
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3895
- local\_iterator
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3783
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3816
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3846
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3876
- locale, 4176
  - std::locale, 3187–3190
- locale\_classes.h, 4176
- locale\_classes.tcc, 4176
- locale\_conv.h, 4177
- locale\_facets.h, 4178
- locale\_facets.tcc, 4179
- locale\_facets\_nonio.h, 4180
- locale\_facets\_nonio.tcc, 4181
- localefwd.h, 4181
- Locales, 130
  - has\_facet, 131
  - use\_facet, 131
- lock
  - std, 712
- log
  - Complex Numbers, 58
- log10
  - Complex Numbers, 58
- logic\_error
  - std::logic\_error, 3203
- lookup\_classname
  - std::regex\_traits< \_Ch\_type >, 3567
- lookup\_collatename
  - std::regex\_traits< \_Ch\_type >, 3568
- losertree.h, 4182
- lower\_bound
  - Binary Search, 44
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3242, 3243
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3351–3353
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3380–3382
  - std::set< \_Key, \_Compare, \_Alloc >, 3610, 3611
- lowest
  - std::numeric\_limits< \_Tp >, 3444
- lu\_counter\_metadata.hpp, 4183
- lu\_map.hpp, 4183
- m\_p\_tbl
  - const\_iterator\_, 1652
  - iterator\_, 1658
- macros.h, 4184
  - \_GLIBCXX\_DEBUG\_VERIFY\_AT, 4188
  - \_\_glibcxx\_check\_erase, 4185
  - \_\_glibcxx\_check\_erase\_after, 4185
  - \_\_glibcxx\_check\_erase\_range, 4185
  - \_\_glibcxx\_check\_erase\_range\_after, 4185
  - \_\_glibcxx\_check\_heap\_pred, 4186
  - \_\_glibcxx\_check\_insert, 4186
  - \_\_glibcxx\_check\_insert\_after, 4186
  - \_\_glibcxx\_check\_insert\_range, 4186
  - \_\_glibcxx\_check\_insert\_range\_after, 4187
  - \_\_glibcxx\_check\_partitioned\_lower, 4187
  - \_\_glibcxx\_check\_partitioned\_lower\_pred, 4187
  - \_\_glibcxx\_check\_partitioned\_upper\_pred, 4188
  - \_\_glibcxx\_check\_sorted\_pred, 4188
- make\_boyer\_moore\_horspool\_searcher
  - experimental/functional, 4135
- make\_boyer\_moore\_searcher
  - experimental/functional, 4135
- make\_default\_searcher
  - experimental/functional, 4135
- make\_error\_code
  - Futures, 100
- make\_error\_condition
  - Futures, 100
- make\_exception\_ptr
  - Exceptions, 87
- make\_heap
  - Heap, 105, 106
- make\_index\_sequence
  - std, 670
- make\_integer\_sequence
  - std, 670
- make\_ostream\_joiner
  - experimental/iterator, 4166
- make\_pair

- Utilities, 441
- make\_reverse\_iterator
  - Iterators, 127
- make\_shared
  - Pointer Abstractions, 305
- make\_signed\_t
  - Metaprogramming, 179
- make\_unique
  - Pointer Abstractions, 306
- make\_unsigned\_t
  - Metaprogramming, 179
- malloc\_allocator.h, 4189
- map, 4189, 4190
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3220–3223
- map.h, 4191
- mapped\_type
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3784
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3817
- mark\_count
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 2460
- mask\_array
  - Numeric Arrays, 247
- mask\_array.h, 4192
- mask\_based\_range\_hashing.hpp, 4193
- match\_any
  - std::regex\_constants, 852
- match\_continuous
  - std::regex\_constants, 852
- match\_default
  - std::regex\_constants, 852
- match\_flag\_type
  - std::regex\_constants, 841
- match\_not\_bol
  - std::regex\_constants, 852
- match\_not\_bow
  - std::regex\_constants, 853
- match\_not\_eol
  - std::regex\_constants, 853
- match\_not\_eow
  - std::regex\_constants, 853
- match\_not\_null
  - std::regex\_constants, 853
- match\_prev\_avail
  - std::regex\_constants, 853
- match\_results
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3256, 3257
- math.h, 4193
- Mathematical Special Functions, 133, 164
  - assoc\_laguerre, 135, 166
  - assoc\_laguerref, 136
  - assoc\_laguerrel, 136
  - assoc\_legendre, 136, 166
  - assoc\_legendref, 137
  - assoc\_legendrel, 138
  - beta, 138, 166
  - betaf, 139
  - betal, 139
  - comp\_ellint\_1, 139, 166
  - comp\_ellint\_1f, 140
  - comp\_ellint\_1l, 140
  - comp\_ellint\_2, 141, 167
  - comp\_ellint\_2f, 142
  - comp\_ellint\_2l, 142
  - comp\_ellint\_3, 142, 167
  - comp\_ellint\_3f, 143
  - comp\_ellint\_3l, 143
  - cyl\_bessel\_i, 143, 167
  - cyl\_bessel\_if, 144
  - cyl\_bessel\_il, 144
  - cyl\_bessel\_j, 145, 167
  - cyl\_bessel\_jf, 145
  - cyl\_bessel\_jl, 146
  - cyl\_bessel\_k, 146, 168
  - cyl\_bessel\_kf, 147
  - cyl\_bessel\_kl, 147
  - cyl\_neumann, 147, 168
  - cyl\_neumannf, 148
  - cyl\_neumannl, 148
  - ellint\_1, 149, 168
  - ellint\_1f, 150
  - ellint\_1l, 150
  - ellint\_2, 150, 168
  - ellint\_2f, 151
  - ellint\_2l, 151
  - ellint\_3, 151, 169
  - ellint\_3f, 152
  - ellint\_3l, 153
  - expint, 153, 169
  - expintf, 154
  - expintl, 154
  - hermite, 154, 169
  - hermitef, 155
  - hermitel, 155
  - laguerre, 155, 169
  - laguerref, 156
  - laguerrel, 156
  - legendre, 156, 170
  - legendref, 157
  - legendrel, 157
  - riemann\_zeta, 158, 170
  - riemann\_zetaf, 158
  - riemann\_zetal, 159
  - sph\_bessel, 159, 170
  - sph\_besself, 160
  - sph\_bessell, 160

- sph\_legendre, [160](#), [170](#)
- sph\_legendref, [161](#)
- sph\_legendrel, [161](#)
- sph\_neumann, [162](#), [171](#)
- sph\_neumannf, [163](#)
- sph\_neumannl, [163](#)
- max
  - \_\_gnu\_parallel, [542](#)
  - Numeric Arrays, [253](#)
  - Sorting, [400](#), [401](#)
  - std::bernoulli\_distribution, [2642](#)
  - std::binomial\_distribution< \_IntType >, [2653](#)
  - std::cauchy\_distribution< \_RealType >, [2675](#)
  - std::chi\_squared\_distribution< \_RealType >, [2683](#)
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2886](#)
  - std::discrete\_distribution< \_IntType >, [2891](#)
  - std::exponential\_distribution< \_RealType >, [2928](#)
  - std::extreme\_value\_distribution< \_RealType >, [2933](#)
  - std::fisher\_f\_distribution< \_RealType >, [2937](#)
  - std::gamma\_distribution< \_RealType >, [2995](#)
  - std::geometric\_distribution< \_IntType >, [3001](#)
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [3042](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [3152](#)
  - std::lognormal\_distribution< \_RealType >, [3212](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [3274](#)
  - std::negative\_binomial\_distribution< \_IntType >, [3391](#)
  - std::normal\_distribution< \_RealType >, [3397](#)
  - std::numeric\_limits< \_Tp >, [3444](#)
  - std::piecewise\_constant\_distribution< \_RealType >, [3508](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [3514](#)
  - std::poisson\_distribution< \_IntType >, [3527](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [3641](#)
  - std::student\_t\_distribution< \_RealType >, [3651](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3662](#)
  - std::uniform\_int\_distribution< \_IntType >, [3755](#)
  - std::uniform\_real\_distribution< \_RealType >, [3759](#)
  - std::weibull\_distribution< \_RealType >, [3959](#)
- max\_bucket\_count
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3806](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3836](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3867](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3895](#)
- max\_count
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, [1569](#)
- max\_digits10
  - std::\_\_numeric\_limits\_base, [1826](#)
  - std::numeric\_limits< \_Tp >, [3448](#)
- max\_element
  - Sorting, [401](#), [402](#)
- max\_element\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1310](#)
- max\_exponent
  - std::\_\_numeric\_limits\_base, [1826](#)
  - std::numeric\_limits< \_Tp >, [3449](#)
- max\_exponent10
  - std::\_\_numeric\_limits\_base, [1827](#)
  - std::numeric\_limits< \_Tp >, [3449](#)
- max\_load\_factor
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3806](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3837](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3867](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3896](#)
- max\_size
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [873](#)
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [937](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1192](#)
  - std::allocator\_traits< \_Alloc >, [1915](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1922](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2531](#)
  - std::deque< \_Tp, \_Alloc >, [2876](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2958](#)
  - std::list< \_Tp, \_Alloc >, [3174](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3244](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [3260](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3353](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3382](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3612](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3723](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3807](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3837](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3868](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3896](#)

- std::vector< \_Tp, \_Alloc >, 3924
- mean
  - std::normal\_distribution< \_RealType >, 3397
  - std::poisson\_distribution< \_IntType >, 3527
- mem\_fn
  - Function Objects, 96
- Memory, 172
- memory, 4193–4195
- memory\_order
  - Atomics, 28
- memory\_resource, 4196
- memoryfwd.h, 4197
- merge
  - Sorting, 402, 403
  - std::forward\_list< \_Tp, \_Alloc >, 2958, 2959
  - std::list< \_Tp, \_Alloc >, 3174, 3175
- merge.h, 4197
- merge\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1310
- merge\_oversampling
  - \_\_gnu\_parallel::Settings, 1310
- mersenne\_twister\_engine
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 3273
- messages
  - std::locale, 3196
  - std::messages< \_CharT >, 3278, 3279
- messages\_members.h, 4198
- metadata\_const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1371
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1377
- metadata\_reference
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 1568
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 1570
- metadata\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1371
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1378
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 1484
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, 1487
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 1568
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 1571
- \_\_gnu\_pbds::sample\_update\_policy, 1602
- Metaprogramming, 173
  - add\_const\_t, 177
  - add\_cv\_t, 177
  - add\_lvalue\_reference\_t, 177
  - add\_pointer\_t, 177
  - add\_rvalue\_reference\_t, 177
  - add\_volatile\_t, 178
  - aligned\_storage\_t, 178
  - alignment\_value, 182
  - common\_type\_t, 178
  - conditional\_t, 178
  - decay\_t, 178
  - enable\_if\_t, 179
  - false\_type, 179
  - make\_signed\_t, 179
  - make\_unsigned\_t, 179
  - remove\_all\_extents\_t, 179
  - remove\_const\_t, 180
  - remove\_cv\_t, 180
  - remove\_extent\_t, 180
  - remove\_pointer\_t, 180
  - remove\_reference\_t, 180
  - remove\_volatile\_t, 181
  - result\_of\_t, 181
  - true\_type, 181
  - underlying\_type\_t, 181
  - void\_t, 181
- microseconds
  - std::chrono, 827
- milliseconds
  - std::chrono, 827
- min
  - \_\_gnu\_parallel, 542
  - Numeric Arrays, 253
  - Sorting, 404
  - std::bernoulli\_distribution, 2642
  - std::binomial\_distribution< \_IntType >, 2653
  - std::cauchy\_distribution< \_RealType >, 2675
  - std::chi\_squared\_distribution< \_RealType >, 2683
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2886
  - std::discrete\_distribution< \_IntType >, 2891
  - std::exponential\_distribution< \_RealType >, 2928
  - std::extreme\_value\_distribution< \_RealType >, 2933
  - std::fisher\_f\_distribution< \_RealType >, 2938
  - std::gamma\_distribution< \_RealType >, 2995
  - std::geometric\_distribution< \_IntType >, 3001
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 3043
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 3153
  - std::lognormal\_distribution< \_RealType >, 3212
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w,

- `__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f` >, [3274](#)
- `std::negative_binomial_distribution< _IntType >`, [3391](#)
- `std::normal_distribution< _RealType >`, [3398](#)
- `std::numeric_limits< _Tp >`, [3445](#)
- `std::piecewise_constant_distribution< _RealType >`, [3508](#)
- `std::piecewise_linear_distribution< _RealType >`, [3514](#)
- `std::poisson_distribution< _IntType >`, [3528](#)
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, [3641](#)
- `std::student_t_distribution< _RealType >`, [3651](#)
- `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, [3662](#)
- `std::uniform_int_distribution< _IntType >`, [3755](#)
- `std::uniform_real_distribution< _RealType >`, [3759](#)
- `std::weibull_distribution< _RealType >`, [3959](#)
- `min_element`
  - Sorting, [405](#)
- `min_element_minimal_n`
  - `__gnu_parallel::Settings`, [1310](#)
- `min_exponent`
  - `std::__numeric_limits_base`, [1827](#)
  - `std::numeric_limits< _Tp >`, [3449](#)
- `min_exponent10`
  - `std::__numeric_limits_base`, [1827](#)
  - `std::numeric_limits< _Tp >`, [3449](#)
- `minmax`
  - Sorting, [406](#)
- `minmax_element`
  - Sorting, [407](#)
- `minstd_rand`
  - Random Number Generators, [318](#)
- `minstd_rand0`
  - Random Number Generators, [318](#)
- `minutes`
  - `std::chrono`, [827](#)
- `mismatch`
  - Non-Mutating, [227–229](#)
- `mod_based_range_hashing.hpp`, [4198](#)
- `modulus`
  - `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, [3156](#)
- `monetary`
  - `std::locale`, [3196](#)
- `money_get`
  - `std::money_get< _CharT, _InIter >`, [3291](#)
- `money_put`
  - `std::money_put< _CharT, _OutIter >`, [3297](#)
- `money_punct`
  - `std::money_punct< _CharT, _Intl >`, [3304](#), [3305](#)
- `move`
  - Mutating, [191](#)
  - Utilities, [441](#)
- `move.h`, [4198](#)
- `move_backward`
  - Mutating, [192](#)
- `move_if_noexcept`
  - Utilities, [442](#)
- `mt19937`
  - Random Number Generators, [318](#)
- `mt19937_64`
  - Random Number Generators, [319](#)
- `mt_allocator.h`, [4199](#)
- `multimap`
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3331–3334](#)
- `multimap.h`, [4200](#), [4201](#)
- `multiplier`
  - `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, [3156](#)
- `multiseq_partition`
  - `__gnu_parallel`, [543](#)
- `multiseq_selection`
  - `__gnu_parallel`, [543](#)
- `multiseq_selection.h`, [4202](#)
- `multiset`
  - `std::multiset< _Key, _Compare, _Alloc >`, [3364–3367](#)
- `multiset.h`, [4202](#), [4203](#)
- `multiway_merge`
  - `__gnu_parallel`, [544](#)
- `multiway_merge.h`, [4204](#)
- `_GLIBCXX_PARALLEL_LENGTH`, [4207](#)
- `multiway_merge_3_variant`
  - `__gnu_parallel`, [546](#)
- `multiway_merge_4_variant`
  - `__gnu_parallel`, [547](#)
- `multiway_merge_exact_splitting`
  - `__gnu_parallel`, [547](#)
- `multiway_merge_loser_tree`
  - `__gnu_parallel`, [548](#)
- `multiway_merge_loser_tree_sentinel`
  - `__gnu_parallel`, [549](#)
- `multiway_merge_loser_tree_unguarded`
  - `__gnu_parallel`, [549](#)
- `multiway_merge_minimal_k`
  - `__gnu_parallel::Settings`, [1311](#)
- `multiway_merge_minimal_n`
  - `__gnu_parallel::Settings`, [1311](#)
- `multiway_merge_oversampling`
  - `__gnu_parallel::Settings`, [1311](#)
- `multiway_merge_sampling_splitting`
  - `__gnu_parallel`, [550](#)
- `multiway_merge_sentinels`
  - `__gnu_parallel`, [551](#)
- `multiway_mergesort.h`, [4207](#)

- Mutating, [183](#)
  - copy, [185](#)
  - copy\_backward, [185](#)
  - copy\_if, [186](#)
  - copy\_n, [186](#)
  - fill, [187](#)
  - fill\_n, [187](#)
  - generate, [189](#)
  - generate\_n, [189](#)
  - is\_partitioned, [190](#)
  - iter\_swap, [191](#)
  - move, [191](#)
  - move\_backward, [192](#)
  - partition, [192](#)
  - partition\_copy, [193](#)
  - partition\_point, [193](#)
  - random\_shuffle, [194](#)
  - remove, [194](#)
  - remove\_copy, [195](#)
  - remove\_copy\_if, [196](#)
  - remove\_if, [196](#)
  - replace, [197](#)
  - replace\_copy\_if, [197](#)
  - replace\_if, [198](#)
  - reverse, [199](#)
  - reverse\_copy, [199](#)
  - rotate, [200](#)
  - rotate\_copy, [200](#)
  - shuffle, [201](#)
  - stable\_partition, [201](#)
  - swap\_ranges, [202](#)
  - transform, [203](#)
  - unique, [204](#)
  - unique\_copy, [205, 206](#)
- mutex, [4208](#)
- Mutexes, [207](#)
  - \_\_cpp\_lib\_shared\_timed\_mutex, [207](#)
  - \_\_shared\_timed\_mutex\_base, [208](#)
  - adopt\_lock, [209](#)
  - defer\_lock, [209](#)
  - swap, [208](#)
  - try\_to\_lock, [209](#)
- name
  - std::locale, [3192](#)
  - std::type\_info, [3749](#)
- nanoseconds
  - std::chrono, [828](#)
- narrow
  - std::\_\_ctype\_abstract\_base<\_CharT>, [1691, 1692](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [2018](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2084](#)
  - std::basic\_ios<\_CharT, \_Traits>, [2130](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2170](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2233](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2288](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [2336](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [2379](#)
  - std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, [2424](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2601](#)
  - std::ctype<\_CharT>, [2764, 2765](#)
  - std::ctype<char>, [2782](#)
  - std::ctype<wchar\_t>, [2800, 2801](#)
  - std::ctype\_byname<\_CharT>, [2817, 2818](#)
  - std::ctype\_byname<char>, [2834](#)
- native\_handle
  - std::thread, [3667](#)
- neg\_format
  - std::moneypunct<\_CharT, \_Intl>, [3312](#)
  - std::moneypunct\_byname<\_CharT, \_Intl>, [3323](#)
- negative\_sign
  - std::moneypunct<\_CharT, \_Intl>, [3312](#)
  - std::moneypunct\_byname<\_CharT, \_Intl>, [3324](#)
- Negators, [210](#)
  - not1, [210](#)
  - not2, [211](#)
- nested\_exception.h, [4209](#)
- new, [4209](#)
  - operator delete, [4210, 4211](#)
  - operator delete[], [4211, 4212](#)
  - operator new, [4212, 4213](#)
  - operator new[], [4213, 4214](#)
- new\_allocator.h, [4215](#)
- new\_handler
  - std, [670](#)
- next\_permutation
  - Sorting, [408](#)
- noboolalpha
  - std, [713](#)
- node.hpp, [4215, 4216](#)
- node\_begin
  - \_\_gnu\_pbds::detail::ov\_tree\_map<Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc>, [1459, 1460](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map<Key, Mapped, Node\_And\_It\_Traits, \_Alloc>, [1492](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map<Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc>, [1503](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map<Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc>, [1512](#)
- node\_const\_iterator
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits<Key, Mapped, Cmp\_Fn, Node\_Update, Node, \_Alloc>, [1381](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits<Key,



- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [1523](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [1525](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [1526](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [1527](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [1529](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [1530](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1535](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1536](#)
- node\_end
  - \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [1460](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, [1493](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [1504](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [1512](#), [1513](#)
- node\_handle.h, [4216](#)
- node\_iterators.hpp, [4216](#), [4217](#)
- node\_metadata\_selector.hpp, [4217](#), [4218](#)
- node\_type
  - \_\_gnu\_pbds::detail::pat\_trie\_base, [1468](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, [1492](#)
- node\_update
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1535](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [1536](#)
- Non-Mutating, [212](#)
  - adjacent\_find, [213](#), [214](#)
  - all\_of, [214](#)
  - any\_of, [215](#)
  - count, [216](#)
  - count\_if, [216](#)
  - equal, [217](#), [218](#)
  - find, [219](#)
  - find\_end, [219](#), [220](#)
  - find\_first\_of, [221](#)
  - find\_if, [222](#)
  - find\_if\_not, [223](#)
  - for\_each, [223](#)
  - is\_permutation, [224](#), [226](#)
  - mismatch, [227–229](#)
  - none\_of, [229](#)
  - search, [230](#), [231](#)
  - search\_n, [231](#), [232](#)
- none
  - std::bitset< \_Nb >, [2665](#)
  - std::locale, [3196](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3724](#)
- none\_of
  - Non-Mutating, [229](#)
- norm
  - Complex Numbers, [58](#)
- Normal Distributions, [233](#)
  - operator!=, [234](#), [235](#)
  - operator<<, [235](#)
  - operator>>, [236](#)
- normal\_distribution
  - std::normal\_distribution< \_RealType >, [3397](#)
- noshowbase
  - std, [713](#)
- noshowpoint
  - std, [713](#)
- noshowpos
  - std, [714](#)
- noskipws
  - std, [714](#)
- nosubs
  - std::regex\_constants, [854](#)
- not1
  - Negators, [210](#)
- not2
  - Negators, [211](#)
- not\_fn
  - experimental/functional, [4135](#)
- notify\_cleared
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1349](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1555](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1590](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1594](#)
- notify\_erase\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1349](#)



\_\_gnu\_pbds::sample\_resize\_policy, 1590  
 \_\_gnu\_pbds::sample\_resize\_trigger, 1594  
 notify\_erase\_search\_end  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1349  
   \_\_gnu\_pbds::sample\_resize\_policy, 1590  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1594  
 notify\_erase\_search\_start  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1349  
   \_\_gnu\_pbds::sample\_resize\_policy, 1590  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1595  
 notify\_erased  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1350  
   \_\_gnu\_pbds::sample\_resize\_policy, 1591  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1595  
 notify\_externally\_resized  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1350  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1595  
 notify\_find\_search\_collision  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1350  
   \_\_gnu\_pbds::sample\_resize\_policy, 1591  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1595  
 notify\_find\_search\_end  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1350  
   \_\_gnu\_pbds::sample\_resize\_policy, 1591  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1595  
 notify\_find\_search\_start  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1351  
   \_\_gnu\_pbds::sample\_resize\_policy, 1591  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1595  
 notify\_insert\_search\_collision  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1351  
   \_\_gnu\_pbds::sample\_resize\_policy, 1591  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1596  
 notify\_insert\_search\_end  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1351  
   \_\_gnu\_pbds::sample\_resize\_policy, 1591  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1596  
 notify\_insert\_search\_start  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1351  
   \_\_gnu\_pbds::sample\_resize\_policy, 1592  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1596  
 notify\_inserted  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1352  
   \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1555  
   \_\_gnu\_pbds::sample\_resize\_policy, 1592  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1596  
 notify\_resized  
   \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1352  
   \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 1556  
   \_\_gnu\_pbds::sample\_range\_hashing, 1585  
   \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 1587  
   \_\_gnu\_pbds::sample\_resize\_policy, 1592  
   \_\_gnu\_pbds::sample\_resize\_trigger, 1596  
 nbuf  
   std, 714  
 nuppercase  
   std, 714  
 npos  
   \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 959  
   \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1200  
   std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2551  
   nth\_element  
     Sorting, 409, 410  
   nth\_element\_minimal\_n  
     \_\_gnu\_parallel::Settings, 1311  
 node  
   node\_metadata.hpp, 4218  
 nullopt  
   Optional values, 287  
 num\_blocks  
   std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3724  
 num\_children  
   \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, 1484  
   \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, 1488  
 num\_get  
   std::num\_get< \_CharT, \_InIter >, 3407  
 num\_put  
   std::num\_put< \_CharT, \_OutIter >, 3428  
 numeric, 4218–4220, 4222  
   std::locale, 3196  
 Numeric Arrays, 237  
   ~gslice, 250  
   apply, 250, 251  
   begin, 251, 252  
   cshift, 252  
   end, 252, 253  
   gslice, 245, 246  
   gslice\_array, 246  
   indirect\_array, 246

- mask\_array, [247](#)
- max, [253](#)
- min, [253](#)
- operator!, [253](#)
- operator<=, [263](#), [264](#)
- operator>=, [271](#), [272](#)
- operator\*=, [257](#), [258](#)
- operator~, [280](#)
- operator^=, [277](#), [278](#)
- operator+, [258](#)
- operator+=, [258–260](#)
- operator-, [260](#)
- operator=, [260](#), [261](#)
- operator/=, [262](#), [263](#)
- operator=, [265–270](#)
- operator%=, [254](#), [255](#)
- operator&=, [255](#), [256](#)
- operator[], [272–276](#)
- operator|=, [278](#), [279](#)
- resize, [280](#)
- shift, [280](#)
- size, [281](#)
- slice, [247](#)
- slice\_array, [248](#)
- start, [281](#), [282](#)
- stride, [282](#)
- sum, [282](#)
- swap, [282](#)
- valarray, [248–250](#)
- numeric\_traits.h, [4223](#)
- numeric\_fwd.h, [4223](#)
- Numerics, [283](#)
- numpunct
  - std::numpunct<\_CharT>, [3471](#), [3472](#)
- oct
  - std, [714](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [2057](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2111](#)
  - std::basic\_ios<\_CharT, \_Traits>, [2143](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2208](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2260](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2316](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [2360](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [2402](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [2448](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2638](#)
  - std::ios\_base, [3073](#)
- off\_type
  - std::basic\_ios<\_CharT, \_Traits>, [2120](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [2467](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3937](#)
- ofstream
  - I/O, [116](#)
- omp\_loop.h, [4225](#)
- omp\_loop\_static.h, [4225](#)
- once\_flag
  - std::once\_flag, [3484](#)
- open
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [987](#), [988](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [1051](#), [1052](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1972](#), [1973](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [2019](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2084](#), [2085](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [2336](#), [2337](#)
- openmode
  - std::basic\_fstream<\_CharT, \_Traits>, [2001](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2067](#)
  - std::basic\_ios<\_CharT, \_Traits>, [2120](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2154](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2218](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2272](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [2325](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [2369](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [2413](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2584](#)
  - std::ios\_base, [3060](#)
- operator\_iterator
  - \_\_gnu\_debug::Safe\_iterator<\_Iterator, \_Sequence>, [1124](#)
  - \_\_gnu\_debug::Safe\_local\_iterator<\_Iterator, \_Sequence>, [1142](#)
- operator\_RAlter
  - \_\_gnu\_parallel::GuardedIterator<\_RAIter, \_Compare>, [1253](#)
- operator\_bool
  - std::basic\_fstream<\_CharT, \_Traits>, [2020](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2085](#)
  - std::basic\_ios<\_CharT, \_Traits>, [2130](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2171](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2234](#)
  - std::basic\_istream<\_CharT, \_Traits>::sentry, [2264](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2289](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [2337](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [2379](#)
  - std::basic\_ostream<\_CharT, \_Traits>::sentry, [2406](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [2424](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2601](#)

- std::function< \_Res(\_ArgTypes...)>, 2978
- std::tr2::bool\_set, 3712
- std::unique\_ptr< \_Tp, \_Dp >, 3767
- std::unique\_ptr< \_Tp[], \_Dp >, 3774
- operator const point\_iterator\_
  - iterator\_, 1655
- operator delete
  - new, 4210, 4211
- operator delete[]
  - new, 4211, 4212
- operator new
  - new, 4212, 4213
- operator new[]
  - new, 4213, 4214
- operator point\_iterator\_
  - iterator\_, 1656
- operator streamoff
  - std::fpos< \_StateT >, 2968
- operator string\_type
  - std::sub\_match< \_Biter >, 3658
- operator!
  - Numeric Arrays, 253
  - std::basic\_fstream< \_CharT, \_Traits >, 2020
  - std::basic\_ifstream< \_CharT, \_Traits >, 2085
  - std::basic\_ios< \_CharT, \_Traits >, 2130
  - std::basic\_iostream< \_CharT, \_Traits >, 2171
  - std::basic\_istream< \_CharT, \_Traits >, 2234
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2289
  - std::basic\_ofstream< \_CharT, \_Traits >, 2337
  - std::basic\_ostream< \_CharT, \_Traits >, 2379
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 2424
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2601
- operator!=
  - \_\_gnu\_cxx, 467, 468
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_iterator, Iterator, \_Alloc >, 1373
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_iterator, Iterator, \_Alloc >, 1379
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 1388
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 1393
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, 1444
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, 1450
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer\_< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 1485
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter\_< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 1488
- \_\_Alloc >, 1488
- Bernoulli Distributions, 37, 38
- Complex Numbers, 59
- const\_iterator\_, 1650
- Dynamic Bitset., 80
- experimental/bits/fs\_path.h, 4126
- iterator\_, 1656
- Iterators, 127
- Normal Distributions, 234, 235
- point\_const\_iterator\_, 1661, 1662
- point\_iterator\_, 1666
- Poisson Distributions, 308, 309
- Random Number Generators, 319–321
- Regular Expressions, 333–337
- std, 715–719
- std::bitset< \_Nb >, 2665
- std::locale, 3192
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3557
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3562
- std::rel\_ops, 855
- Uniform Distributions, 429, 430
- Utilities, 442
- operator<
  - \_\_gnu\_cxx, 471, 472
  - \_\_gnu\_parallel::\_GuardedIterator< \_RAIter, \_Compare >, 1254
  - experimental/bits/fs\_path.h, 4126
  - Regular Expressions, 337, 339–341
  - std, 722, 724–730
  - Utilities, 442
- operator<<
  - Bernoulli Distributions, 38, 39
  - Complex Numbers, 64
  - Dynamic Bitset., 81
  - experimental/bits/fs\_path.h, 4126
  - Normal Distributions, 235
  - Pointer Abstractions, 306
  - Poisson Distributions, 310, 311
  - Random Number Generators, 322
  - Regular Expressions, 341
  - std, 730–738
  - std::\_detail, 800, 801
  - std::basic\_fstream< \_CharT, \_Traits >, 2020–2028
  - std::basic\_ifstream< \_CharT, \_Traits >, 2171–2173, 2175–2179
  - std::basic\_iostream< \_CharT, \_Traits >, 2338–2345
  - std::basic\_ostream< \_CharT, \_Traits >, 2380–2387
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 2425–2432
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2602–2609
  - std::binomial\_distribution< \_IntType >, 2656

- std::bitset< \_Nb >, 2666
- std::chi\_squared\_distribution< \_RealType >, 2684
- std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2888
- std::discrete\_distribution< \_IntType >, 2893
- std::fisher\_f\_distribution< \_RealType >, 2939
- std::gamma\_distribution< \_RealType >, 2997
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 3154
- std::lognormal\_distribution< \_RealType >, 3213
- std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 3274
- std::negative\_binomial\_distribution< \_IntType >, 3393
- std::normal\_distribution< \_RealType >, 3400
- std::piecewise\_constant\_distribution< \_RealType >, 3509
- std::piecewise\_linear\_distribution< \_RealType >, 3515
- std::poisson\_distribution< \_IntType >, 3529
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3642
- std::student\_t\_distribution< \_RealType >, 3653
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3663
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3725
- Uniform Distributions, 430
- operator<=>
  - Numeric Arrays, 263, 264
  - std::bitset< \_Nb >, 2666
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3725
- operator<=
  - \_\_gnu\_cxx, 472, 473
  - \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 1255
  - Dynamic Bitset., 81
  - experimental/bits/fs\_path.h, 4126
  - Regular Expressions, 342–345
  - std, 738–742
  - std::\_\_debug, 796
  - std::\_\_profile, 824
  - std::rel\_ops, 856
  - Utilities, 443
- operator>
  - \_\_gnu\_cxx, 476, 477
  - Dynamic Bitset., 81
  - experimental/bits/fs\_path.h, 4127
  - Regular Expressions, 349–352
  - std, 752–755
  - std::\_\_debug, 796
  - std::\_\_profile, 824
  - std::rel\_ops, 856
  - Utilities, 443
- operator>>
  - Bernoulli Distributions, 39, 40
  - Complex Numbers, 66
  - Dynamic Bitset., 82
  - experimental/bits/fs\_path.h, 4127
  - Normal Distributions, 236
  - Poisson Distributions, 311, 312
  - std, 760–766
  - std::\_\_detail, 801
  - std::basic\_fstream< \_CharT, \_Traits >, 2028–2036
  - std::basic\_ifstream< \_CharT, \_Traits >, 2086–2093
  - std::basic\_iostream< \_CharT, \_Traits >, 2180–2187
  - std::basic\_istream< \_CharT, \_Traits >, 2234–2236, 2238–2242
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2289–2291, 2293–2297
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2609–2617
  - std::binomial\_distribution< \_IntType >, 2656
  - std::bitset< \_Nb >, 2667
  - std::chi\_squared\_distribution< \_RealType >, 2685
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2888
  - std::discrete\_distribution< \_IntType >, 2893
  - std::fisher\_f\_distribution< \_RealType >, 2940
  - std::gamma\_distribution< \_RealType >, 2998
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 3045
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 3155
  - std::lognormal\_distribution< \_RealType >, 3214
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 3275
  - std::negative\_binomial\_distribution< \_IntType >, 3393
  - std::normal\_distribution< \_RealType >, 3400
  - std::piecewise\_constant\_distribution< \_RealType >, 3510
  - std::piecewise\_linear\_distribution< \_RealType >, 3516
  - std::poisson\_distribution< \_IntType >, 3530
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3643
  - std::student\_t\_distribution< \_RealType >, 3653
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3664
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3726
  - Uniform Distributions, 431
- operator>>=
  - Numeric Arrays, 271, 272
  - std::bitset< \_Nb >, 2667
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3726
- operator>=

- `__gnu_cxx`, 477, 478
- Dynamic Bitset., 82
- experimental/bits/fs\_path.h, 4127
- Regular Expressions, 352, 353, 355, 356
- std, 756–759
- std::\_\_debug, 797
- std::\_\_profile, 824
- std::rel\_ops, 857
- Utilities, 443
- operator\*
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1124
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1142
  - `__gnu_parallel::GuardedIterator< _RAIter, _Compare >`, 1254
  - `__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 1373
  - `__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 1379
  - `__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 1388
  - `__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 1393
  - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >`, 1444
  - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >`, 1450
  - `__gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >`, 1465
  - `__gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`, 1485
  - `__gnu_pbds::detail::pat_trie_base::Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`, 1489
- Complex Numbers, 59, 60
- const\_iterator\_, 1650
- iterator\_, 1656
- point\_const\_iterator\_, 1662
- point\_iterator\_, 1666
- std::auto\_ptr< \_Tp >, 1951
- std::back\_insert\_iterator< \_Container >, 1957
- std::front\_insert\_iterator< \_Container >, 2973
- std::insert\_iterator< \_Container >, 3051
- std::istreambuf\_iterator< \_CharT, \_Traits >, 3140
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 3493
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3557
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3563
- std::reverse\_iterator< \_Iterator >, 3578
- std::unique\_ptr< \_Tp, \_Dp >, 3768
- operator\*=
  - Complex Numbers, 60
  - Numeric Arrays, 257, 258
- operator~
  - Numeric Arrays, 280
  - std::bitset< \_Nb >, 2669
  - std::regex\_constants, 848
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3728
- operator^
  - Dynamic Bitset., 82
  - std, 767
  - std::regex\_constants, 845
- operator^=
  - Numeric Arrays, 277, 278
  - std::bitset< \_Nb >, 2669
  - std::regex\_constants, 846
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3728
- `__gnu_cxx::subtractive_rng`, 1095
- `__gnu_parallel::Nothing`, 1289
- `__gnu_parallel::RandomNumber`, 1301, 1302
- `__gnu_parallel::accumulate_selector< _It >`, 1202
- `__gnu_parallel::adjacent_find_selector`, 1205
- `__gnu_parallel::count_if_selector< _It, _Diff >`, 1210
- `__gnu_parallel::count_selector< _It, _Diff >`, 1212
- `__gnu_parallel::fill_selector< _It >`, 1213
- `__gnu_parallel::find_first_of_selector< _Filterator >`, 1216
- `__gnu_parallel::find_if_selector`, 1217
- `__gnu_parallel::for_each_selector< _It >`, 1219
- `__gnu_parallel::generate_selector< _It >`, 1220
- `__gnu_parallel::identity_selector< _It >`, 1225
- `__gnu_parallel::inner_product_selector< _It, _It2, _Tp >`, 1227
- `__gnu_parallel::mismatch_selector`, 1231
- `__gnu_parallel::replace_if_selector< _It, _Op, _Tp >`, 1236
- `__gnu_parallel::replace_selector< _It, _Tp >`, 1238
- `__gnu_parallel::transform1_selector< _It >`, 1240
- `__gnu_parallel::transform2_selector< _It >`, 1242
- `__gnu_pbds::direct_mask_range_hashing< Size_Type >`, 1543
- `__gnu_pbds::direct_mod_range_hashing< Size_Type >`, 1544
- `__gnu_pbds::linear_probe_fn< Size_Type >`, 1565
- `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`, 1569
- `__gnu_pbds::lu_move_to_front_policy< _Alloc >`, 1571
- `__gnu_pbds::quadratic_probe_fn< Size_Type >`, 1579
- `__gnu_pbds::sample_probe_fn`, 1584
- `__gnu_pbds::sample_range_hashing`, 1586

- `__gnu_pbds::sample_ranged_hash_fn`, 1587
- `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 1601
- `__gnu_pbds::sample_update_policy`, 1603
- `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`, 1612
- `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 1618
- `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 1622
- `std::bernoulli_distribution`, 2642
- `std::binomial_distribution< _IntType >`, 2654
- `std::cauchy_distribution< _RealType >`, 2675
- `std::chi_squared_distribution< _RealType >`, 2683
- `std::default_delete< _Tp >`, 2848
- `std::default_delete< _Tp[] >`, 2850
- `std::discard_block_engine< _RandomNumberEngine, __p, __r >`, 2886
- `std::discrete_distribution< _IntType >`, 2891
- `std::exponential_distribution< _RealType >`, 2928
- `std::extreme_value_distribution< _RealType >`, 2934
- `std::fisher_f_distribution< _RealType >`, 2938
- `std::function< _Res(_ArgTypes...) >`, 2978
- `std::gamma_distribution< _RealType >`, 2995, 2996
- `std::geometric_distribution< _IntType >`, 3001
- `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`, 3043
- `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, 3153
- `std::locale`, 3192
- `std::lognormal_distribution< _RealType >`, 3212
- `std::negative_binomial_distribution< _IntType >`, 3391
- `std::normal_distribution< _RealType >`, 3398
- `std::piecewise_constant_distribution< _RealType >`, 3508
- `std::piecewise_linear_distribution< _RealType >`, 3514
- `std::poisson_distribution< _IntType >`, 3528
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 3641
- `std::student_t_distribution< _RealType >`, 3651
- `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, 3662
- `std::uniform_int_distribution< _IntType >`, 3755
- `std::uniform_real_distribution< _RealType >`, 3760
- `std::weibull_distribution< _RealType >`, 3959
- `operator+`
  - `__gnu_cxx`, 468–470
  - Complex Numbers, 61
  - Numeric Arrays, 258
  - `std`, 720–722
  - `std::fpos< _StateT >`, 2968
  - `std::reverse_iterator< _Iterator >`, 3579
- `operator++`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1124
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1142
  - `__gnu_parallel::GuardedIterator< _RAIter, _Compare >`, 1254
  - `const_iterator_`, 1651
  - `iterator_`, 1656, 1657
  - `std::back_insert_iterator< _Container >`, 1957
  - `std::front_insert_iterator< _Container >`, 2973
  - `std::insert_iterator< _Container >`, 3052
  - `std::istreambuf_iterator< _CharT, _Traits >`, 3141
  - `std::ostreambuf_iterator< _CharT, _Traits >`, 3494
  - `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3557, 3558
  - `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3563
  - `std::reverse_iterator< _Iterator >`, 3579
- `operator+=`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 937, 939, 940
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1192
  - Complex Numbers, 62
  - Numeric Arrays, 258–260
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 2531–2533
  - `std::complex< _Tp >`, 2742
  - `std::fpos< _StateT >`, 2968
  - `std::reverse_iterator< _Iterator >`, 3580
- `operator-`
  - Complex Numbers, 62, 63
  - Dynamic Bitset., 80
  - Numeric Arrays, 260
  - `std::fpos< _StateT >`, 2969
  - `std::reverse_iterator< _Iterator >`, 3580
- `operator->`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence >`, 1125
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1143
  - `__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 1388
  - `__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 1393
  - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >`, 1445
  - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >`, 1450
  - `const_iterator_`, 1651
  - `iterator_`, 1657
  - `point_const_iterator_`, 1662
  - `point_iterator_`, 1666



- std::auto\_ptr< \_Tp >, [1951](#)
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3558](#)
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3563](#)
- std::reverse\_iterator< \_Iterator >, [3581](#)
- std::unique\_ptr< \_Tp, \_Dp >, [3768](#)
- operator--
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1125](#)
  - std::reverse\_iterator< \_Iterator >, [3580](#), [3581](#)
- operator-=
  - Complex Numbers, [63](#)
  - Numeric Arrays, [260](#), [261](#)
  - std::complex< \_Tp >, [2743](#)
  - std::fpos< \_StateT >, [2969](#)
  - std::reverse\_iterator< \_Iterator >, [3581](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3725](#)
- operator/
  - Complex Numbers, [63](#), [64](#)
  - experimental/bits/fs\_path.h, [4126](#)
- operator/=
  - Complex Numbers, [64](#)
  - Numeric Arrays, [262](#), [263](#)
- operator=
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [940–942](#)
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence >, [1126](#)
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, [1143](#)
  - Complex Numbers, [65](#)
  - Numeric Arrays, [265–270](#)
  - std::\_\_allocated\_ptr< \_Alloc >, [1669](#)
  - std::auto\_ptr< \_Tp >, [1951](#), [1952](#)
  - std::back\_insert\_iterator< \_Container >, [1957](#)
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, [2460–2462](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2533–2535](#)
  - std::deque< \_Tp, \_Alloc >, [2876](#), [2877](#)
  - std::experimental::fundamentals\_v1::any, [2912](#), [2913](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2959](#), [2960](#)
  - std::front\_insert\_iterator< \_Container >, [2973](#)
  - std::function< \_Res(\_ArgTypes...)>, [2979–2981](#)
  - std::insert\_iterator< \_Container >, [3052](#)
  - std::list< \_Tp, \_Alloc >, [3175](#), [3176](#)
  - std::locale, [3193](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3244](#), [3245](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [3261](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3353](#), [3354](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3382](#), [3383](#)
  - std::once\_flag, [3484](#)
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3489](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3494](#)
  - std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3558](#)
  - std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3564](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3612](#), [3613](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3726](#)
  - std::unique\_ptr< \_Tp, \_Dp >, [3768](#), [3769](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3775](#), [3776](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3807](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3838](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3868](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3897](#)
  - std::vector< \_Tp, \_Alloc >, [3924](#), [3925](#)
  - operator==
    - \_\_gnu\_cxx, [474](#), [475](#)
    - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_iterator, Iterator, \_Alloc >, [1373](#)
    - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_iterator, Iterator, \_Alloc >, [1380](#)
    - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1389](#)
    - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [1394](#)
    - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [1445](#)
    - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [1451](#)
    - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [1485](#)
    - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [1489](#)
    - Complex Numbers, [65](#), [66](#)
    - const\_iterator\_, [1651](#)
    - experimental/bits/fs\_path.h, [4127](#)
    - iterator\_, [1657](#)
    - Iterators, [128](#)
    - point\_const\_iterator\_, [1662](#)
    - point\_iterator\_, [1666](#), [1667](#)
    - Regular Expressions, [345–348](#)
    - std, [742–745](#), [747](#), [749–752](#)
    - std::bernoulli\_distribution, [2644](#)
    - std::binomial\_distribution< \_IntType >, [2656](#)
    - std::bitset< \_Nb >, [2667](#)
    - std::cauchy\_distribution< \_RealType >, [2676](#)
    - std::chi\_squared\_distribution< \_RealType >, [2685](#)

- std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2888](#)
- std::discrete\_distribution< \_IntType >, [2893](#)
- std::exponential\_distribution< \_RealType >, [2929](#)
- std::extreme\_value\_distribution< \_RealType >, [2935](#)
- std::fisher\_f\_distribution< \_RealType >, [2939](#)
- std::gamma\_distribution< \_RealType >, [2997](#)
- std::geometric\_distribution< \_IntType >, [3002](#)
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [3044](#)
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [3155](#)
- std::locale, [3193](#)
- std::lognormal\_distribution< \_RealType >, [3214](#)
- std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [3275](#)
- std::negative\_binomial\_distribution< \_IntType >, [3393](#)
- std::normal\_distribution< \_RealType >, [3400](#)
- std::piecewise\_constant\_distribution< \_RealType >, [3510](#)
- std::piecewise\_linear\_distribution< \_RealType >, [3516](#)
- std::poisson\_distribution< \_IntType >, [3530](#)
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3558](#)
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3564](#)
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [3643](#)
- std::student\_t\_distribution< \_RealType >, [3653](#)
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3664](#)
- std::uniform\_int\_distribution< \_IntType >, [3757](#)
- std::uniform\_real\_distribution< \_RealType >, [3761](#)
- std::weibull\_distribution< \_RealType >, [3960](#)
- Utilities, [443](#)
- operator%=
  - Numeric Arrays, [254](#), [255](#)
- operator&
  - Dynamic Bitset., [80](#)
  - std, [720](#)
  - std::regex\_constants, [844](#)
- operator&=
  - Numeric Arrays, [255](#), [256](#)
  - std::bitset< \_Nb >, [2666](#)
  - std::regex\_constants, [844](#), [845](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3724](#), [3725](#)
- operator[]
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [942](#), [943](#)
  - Numeric Arrays, [272–276](#)
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2535](#)
- std::bitset< \_Nb >, [2668](#)
- std::deque< \_Tp, \_Alloc >, [2877](#), [2878](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3245](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [3261](#)
- std::reverse\_iterator< \_Iterator >, [3582](#)
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3727](#)
- std::unique\_ptr< \_Tp[], \_Dp >, [3776](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3808](#)
- std::vector< \_Tp, \_Alloc >, [3925](#), [3926](#)
- operator|
  - Dynamic Bitset., [83](#)
  - std, [767](#)
  - std::regex\_constants, [846](#), [847](#)
- operator|=
  - Numeric Arrays, [278](#), [279](#)
  - std::bitset< \_Nb >, [2669](#)
  - std::regex\_constants, [847](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3728](#)
- opt\_random.h, [4226](#)
- optimize
  - std::regex\_constants, [854](#)
- optional, [4226](#)
  - \_\_constexpr\_addressof, [4229](#)
- Optional values, [284](#)
  - \_\_constexpr\_addressof, [287](#)
  - in\_place, [287](#)
  - nullopt, [287](#)
- order\_of\_key
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_Cltr, Node\_ltr, Cmp\_Fn, \_Alloc >, [1612](#)
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_ltr, \_ATraits, \_Alloc >, [1618](#)
- order\_of\_prefix
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_ltr, \_ATraits, \_Alloc >, [1619](#)
- order\_preserving
  - \_\_gnu\_pbds::container\_traits< Cntnr >, [1362](#)
- order\_statistics\_imp.hpp, [4229](#)
- ordered\_base.h, [4229](#)
- os\_defines.h, [4230](#)
- ostream, [4230](#)
  - I/O, [116](#)
- ostream.tcc, [4232](#)
- ostream\_insert.h, [4232](#)
- ostream\_iterator
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3488](#), [3489](#)
- ostream\_type
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3487](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3491](#)
- ostreambuf\_iterator
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3493](#)



- ostreamstream
  - I/O, [116](#)
- out
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, [1678](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2057](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2111](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2143](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2208](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2260](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2316](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2360](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2403](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2448](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2638](#)
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, [2694](#)
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, [2699](#)
  - std::codecvt< char, char, mbstate\_t >, [2704](#)
  - std::codecvt< char16\_t, char, mbstate\_t >, [2709](#)
  - std::codecvt< char32\_t, char, mbstate\_t >, [2714](#)
  - std::codecvt< wchar\_t, char, mbstate\_t >, [2719](#)
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, [2725](#)
  - std::ios\_base, [3074](#)
- ov\_tree\_map\_.hpp, [4233](#)
- overflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [989](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1053](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1077](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1974](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2472](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2558](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3942](#)
- p
  - std::bernoulli\_distribution, [2643](#)
  - std::binomial\_distribution< \_IntType >, [2654](#)
  - std::geometric\_distribution< \_IntType >, [3001](#)
  - std::negative\_binomial\_distribution< \_IntType >, [3392](#)
- pair
  - std::pair< \_T1, \_T2 >, [3504](#)
- pairing\_heap\_.hpp, [4233](#)
- par\_loop.h, [4234](#)
- parallel.h, [4234](#)
- parallel\_balanced
  - \_\_gnu\_parallel, [504](#)
- parallel\_multiway\_merge
  - \_\_gnu\_parallel, [553](#)
- parallel\_omp\_loop
  - \_\_gnu\_parallel, [504](#)
- parallel\_omp\_loop\_static
  - \_\_gnu\_parallel, [504](#)
- parallel\_sort\_mwms
  - \_\_gnu\_parallel, [553](#)
- parallel\_sort\_mwms\_pu
  - \_\_gnu\_parallel, [554](#)
- parallel\_tag
  - \_\_gnu\_parallel::parallel\_tag, [1334](#)
- parallel\_taskqueue
  - \_\_gnu\_parallel, [504](#)
- parallel\_unbalanced
  - \_\_gnu\_parallel, [504](#)
- param
  - std::bernoulli\_distribution, [2643](#)
  - std::binomial\_distribution< \_IntType >, [2654](#), [2655](#)
  - std::cauchy\_distribution< \_RealType >, [2675](#), [2676](#)
  - std::chi\_squared\_distribution< \_RealType >, [2683](#), [2684](#)
  - std::discrete\_distribution< \_IntType >, [2891](#), [2892](#)
  - std::exponential\_distribution< \_RealType >, [2928](#), [2929](#)
  - std::extreme\_value\_distribution< \_RealType >, [2934](#)
  - std::fisher\_f\_distribution< \_RealType >, [2938](#)
  - std::gamma\_distribution< \_RealType >, [2996](#)
  - std::geometric\_distribution< \_IntType >, [3001](#), [3002](#)
  - std::lognormal\_distribution< \_RealType >, [3212](#), [3213](#)
  - std::negative\_binomial\_distribution< \_IntType >, [3392](#)
  - std::normal\_distribution< \_RealType >, [3398](#), [3399](#)
  - std::piecewise\_constant\_distribution< \_RealType >, [3508](#), [3509](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [3514](#), [3515](#)
  - std::poisson\_distribution< \_IntType >, [3528](#), [3529](#)
  - std::student\_t\_distribution< \_RealType >, [3652](#)
  - std::uniform\_int\_distribution< \_IntType >, [3756](#)
  - std::uniform\_real\_distribution< \_RealType >, [3760](#)
  - std::weibull\_distribution< \_RealType >, [3959](#), [3960](#)
- parse\_numbers.h, [4234](#)
- partial\_sort
  - Sorting, [410](#), [411](#)
- partial\_sort\_copy
  - Sorting, [411](#), [412](#)
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1311](#)
- partial\_sum
  - std, [768](#), [769](#)
- partial\_sum.h, [4235](#)
- partial\_sum\_dilation
  - \_\_gnu\_parallel::Settings, [1312](#)
- partial\_sum\_minimal\_n

- \_\_gnu\_parallel::Settings, [1312](#)
- partition
  - Mutating, [192](#)
- partition.h, [4235](#)
  - \_GLIBCXX\_VOLATILE, [4236](#)
- partition\_chunk\_share
  - \_\_gnu\_parallel::Settings, [1312](#)
- partition\_chunk\_size
  - \_\_gnu\_parallel::Settings, [1312](#)
- partition\_copy
  - Mutating, [193](#)
- partition\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1312](#)
- partition\_point
  - Mutating, [193](#)
- pat\_trie.hpp, [4236](#)
- pat\_trie\_base.hpp, [4237](#)
- pbackfail
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [989](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [1053](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf<\_CharT, \_Traits>, [1078](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1974](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [2472](#)
  - std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>, [2559](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3942](#)
- pbase
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [991](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [1055](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf<\_CharT, \_Traits>, [1078](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1976](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [2473](#)
  - std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>, [2559](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3943](#)
- pbump
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [991](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [1055](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf<\_CharT, \_Traits>, [1079](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1976](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [2473](#)
  - std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>, [2560](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3943](#)
- peek
  - std::basic\_fstream<\_CharT, \_Traits>, [2036](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2093](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2188](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2243](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2298](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2617](#)
- perms
  - Filesystem TS, [94](#)
- piecewise\_construct
  - Utilities, [446](#)
- pod\_char\_traits.h, [4237](#)
- point\_const\_iterator.hpp, [4238](#), [4239](#)
- point\_const\_iterator\_, [1658](#)
  - const\_pointer, [1659](#)
  - const\_reference, [1659](#)
  - difference\_type, [1660](#)
  - iterator\_category, [1660](#)
  - operator!=, [1661](#), [1662](#)
  - operator\*, [1662](#)
  - operator->, [1662](#)
  - operator==, [1662](#)
  - point\_const\_iterator\_, [1661](#)
  - pointer, [1660](#)
  - reference, [1660](#)
  - value\_type, [1660](#)
- point\_iterator.hpp, [4239](#)
- point\_iterator\_, [1663](#)
  - const\_pointer, [1664](#)
  - const\_reference, [1664](#)
  - difference\_type, [1664](#)
  - iterator\_category, [1664](#)
  - operator!=, [1666](#)
  - operator\*, [1666](#)
  - operator->, [1666](#)
  - operator==, [1666](#), [1667](#)
  - point\_iterator\_, [1665](#)
  - pointer, [1664](#)
  - reference, [1665](#)
  - value\_type, [1665](#)
- point\_iterators.hpp, [4239](#)
- pointer
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator<Value\_Type, Entry, Simple, \_Alloc>, [1386](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator<Value\_Type, Entry, Simple, \_Alloc>, [1392](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator<Node, \_Alloc>, [1443](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator<Node, \_Alloc>, [1449](#)
  - const\_iterator\_, [1649](#)
  - iterator\_, [1654](#)
  - point\_const\_iterator\_, [1660](#)
  - point\_iterator\_, [1664](#)
  - std::allocator\_traits<\_Alloc>, [1911](#)
  - std::allocator\_traits<allocator<\_Tp>>, [1918](#)
  - std::back\_insert\_iterator<\_Container>, [1956](#)
  - std::front\_insert\_iterator<\_Container>, [2972](#)
  - std::insert\_iterator<\_Container>, [3050](#)
  - std::istream\_iterator<\_Tp, \_CharT, \_Traits, \_Dist>, [3134](#)
  - std::istreambuf\_iterator<\_CharT, \_Traits>, [3138](#)

- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, [3142](#)
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3487](#)
- std::ostreambuf\_iterator< \_CharT, \_Traits >, [3492](#)
- std::pointer\_traits< \_Ptr >, [3523](#)
- std::pointer\_traits< \_Tp \* >, [3525](#)
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, [3551](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3592](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3784](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3817](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3846](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3876](#)
- Pointer Abstractions, [288](#)
  - allocate\_shared, [292](#)
  - atomic\_compare\_exchange\_strong, [292](#), [293](#)
  - atomic\_compare\_exchange\_strong\_explicit, [293](#), [295](#)
  - atomic\_compare\_exchange\_weak, [296](#)
  - atomic\_compare\_exchange\_weak\_explicit, [297](#)
  - atomic\_exchange, [298](#), [299](#)
  - atomic\_exchange\_explicit, [299](#), [300](#)
  - atomic\_is\_lock\_free, [300](#), [301](#)
  - atomic\_load, [301](#), [302](#)
  - atomic\_load\_explicit, [302](#), [303](#)
  - atomic\_store, [303](#), [304](#)
  - atomic\_store\_explicit, [304](#), [305](#)
  - get\_deleter, [305](#)
  - make\_shared, [305](#)
  - make\_unique, [306](#)
  - operator<=, [306](#)
- pointer.h, [4240](#)
- pointer\_to
  - std::pointer\_traits< \_Tp \* >, [3525](#)
- Poisson Distributions, [307](#)
  - operator!=, [308](#), [309](#)
  - operator<, [310](#), [311](#)
  - operator>, [311](#), [312](#)
- polar
  - Complex Numbers, [66](#)
- Policy-Based Data Structures, [314](#)
- policy\_access\_fn\_imps.hpp, [4242](#), [4243](#)
- pool\_allocator.h, [4243](#)
- pop
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3534](#)
  - std::queue< \_Tp, \_Sequence >, [3541](#)
  - std::stack< \_Tp, \_Sequence >, [3648](#)
- pop\_back
  - \_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [943](#)
  - \_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Tp >, [1303](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2536](#)
  - std::deque< \_Tp, \_Alloc >, [2878](#)
  - std::list< \_Tp, \_Alloc >, [3177](#)
  - std::vector< \_Tp, \_Alloc >, [3926](#)
- pop\_front
  - \_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Tp >, [1303](#)
  - std::deque< \_Tp, \_Alloc >, [2878](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2960](#)
  - std::list< \_Tp, \_Alloc >, [3177](#)
- pop\_heap
  - Heap, [106](#), [107](#)
- pos\_format
  - std::moneypunct< \_CharT, \_Intl >, [3313](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3324](#)
- pos\_type
  - std::basic\_ios< \_CharT, \_Traits >, [2121](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2467](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3938](#)
- position
  - std::match\_results< \_Bi\_iter, \_Alloc >, [3262](#)
- positive\_sign
  - std::moneypunct< \_CharT, \_Intl >, [3313](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3325](#)
- postypes.h, [4243](#)
- pow
  - Complex Numbers, [66](#), [67](#)
- power
  - SGL, [380](#), [381](#)
- pptr
  - \_gnu\_cxx::enc\_filebuf< \_CharT >, [992](#)
  - \_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1056](#)
  - \_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1079](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1977](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2474](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2560](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3944](#)
- precision
  - std::basic\_fstream< \_CharT, \_Traits >, [2037](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2094](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2131](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2188](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2243](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2298](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2345](#), [2346](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2387](#), [2388](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2432](#), [2433](#)

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2618
- std::ios\_base, 3064
- predefined\_ops.h, 4244
- prefix
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3262
- prefix\_range
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 1623
- prefix\_search\_node\_update\_imp.hpp, 4245
- prev\_permutation
  - Sorting, 413
- priority\_queue
  - Heap-Based, 111
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 3533
- priority\_queue.hpp, 4245
- priority\_queue\_base\_dispatch.hpp, 4246
- probabilities
  - std::discrete\_distribution< \_IntType >, 2892
- probe\_fn\_base.hpp, 4246
- profiler.h, 4247
- profiler\_algos.h, 4249
- profiler\_container\_size.h, 4249
- profiler\_hash\_func.h, 4250
- profiler\_hashtable\_size.h, 4250
- profiler\_list\_to\_slist.h, 4251
- profiler\_list\_to\_vector.h, 4251
- profiler\_map\_to\_unordered\_map.h, 4252
- profiler\_node.h, 4252
- profiler\_state.h, 4253
- profiler\_trace.h, 4254
- profiler\_vector\_size.h, 4256
- profiler\_vector\_to\_list.h, 4256
- propagate\_const, 4257
- propagate\_on\_container\_copy\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 867
  - std::allocator\_traits< \_Alloc >, 1911
  - std::allocator\_traits< allocator< \_Tp > >, 1918
- propagate\_on\_container\_move\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 867
  - std::allocator\_traits< \_Alloc >, 1911
  - std::allocator\_traits< allocator< \_Tp > >, 1918
- propagate\_on\_container\_swap
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 868
  - std::allocator\_traits< \_Alloc >, 1911
  - std::allocator\_traits< allocator< \_Tp > >, 1918
- ptr\_fun
  - Adaptors for pointers to functions, 6
- ptr\_traits.h, 4259
- pubimbue
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 992
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1056
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1079
- std::basic\_filebuf< \_CharT, \_Traits >, 1977
- std::basic\_streambuf< \_CharT, \_Traits >, 2474
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2560
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3944
- pubseekoff
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 992
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1056
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1080
- std::basic\_filebuf< \_CharT, \_Traits >, 1977
- std::basic\_streambuf< \_CharT, \_Traits >, 2474
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2561
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3944
- pubseekpos
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 993
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1057
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1080
- std::basic\_filebuf< \_CharT, \_Traits >, 1978
- std::basic\_streambuf< \_CharT, \_Traits >, 2475
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2561
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3945
- pubsetbuf
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 993
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1057
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1082
- std::basic\_filebuf< \_CharT, \_Traits >, 1978
- std::basic\_streambuf< \_CharT, \_Traits >, 2475
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2563
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3945
- pubsync
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 994
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1058
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1082
- std::basic\_filebuf< \_CharT, \_Traits >, 1979
- std::basic\_streambuf< \_CharT, \_Traits >, 2476
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2563
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3945
- push
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 3535
  - std::queue< \_Tp, \_Sequence >, 3541
  - std::stack< \_Tp, \_Sequence >, 3648
- push\_back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 944
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2536
  - std::deque< \_Tp, \_Alloc >, 2878
  - std::list< \_Tp, \_Alloc >, 3177
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3729

- std::vector< \_Tp, \_Alloc >, 3926
- push\_front
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue<queue.h, 4260
  - \_Tp >, 1304
- std::deque< \_Tp, \_Alloc >, 2879
- std::forward\_list< \_Tp, \_Alloc >, 2961
- std::list< \_Tp, \_Alloc >, 3178
- push\_heap
  - Heap, 107, 108
- put
  - std::basic\_fstream< \_CharT, \_Traits >, 2038
  - std::basic\_istream< \_CharT, \_Traits >, 2189
  - std::basic\_ofstream< \_CharT, \_Traits >, 2346
  - std::basic\_ostream< \_CharT, \_Traits >, 2388
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2433
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2619
  - std::money\_put< \_CharT, \_Outiter >, 3299, 3300
  - std::num\_put< \_CharT, \_Outiter >, 3434–3441
  - std::time\_put< \_CharT, \_Outiter >, 3699, 3700
  - std::time\_put\_byname< \_CharT, \_Outiter >, 3704
- put\_money
  - std, 769
- put\_time
  - std, 770
- putback
  - std::basic\_fstream< \_CharT, \_Traits >, 2038
  - std::basic\_ifstream< \_CharT, \_Traits >, 2095
  - std::basic\_istream< \_CharT, \_Traits >, 2189
  - std::basic\_istream< \_CharT, \_Traits >, 2244
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2299
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2619
- pword
  - std::basic\_fstream< \_CharT, \_Traits >, 2039
  - std::basic\_ifstream< \_CharT, \_Traits >, 2095
  - std::basic\_ios< \_CharT, \_Traits >, 2131
  - std::basic\_istream< \_CharT, \_Traits >, 2190
  - std::basic\_istream< \_CharT, \_Traits >, 2244
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2299
  - std::basic\_ofstream< \_CharT, \_Traits >, 2347
  - std::basic\_ostream< \_CharT, \_Traits >, 2389
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2434
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2620
  - std::ios\_base, 3065
- qsb\_steals
  - \_\_gnu\_parallel::Settings, 1313
- quadratic\_probe\_fn\_imp.hpp, 4260
- queue, 4260
  - std::queue< \_Tp, \_Sequence >, 3539
- queue.h, 4260
  - \_GLIBCXX\_VOLATILE, 4261
- quicksort.h, 4261
- quiet\_NaN
  - std::numeric\_limits< \_Tp >, 3445
- quoted
  - std, 770
- quoted\_string.h, 4261
- r\_erase\_fn\_imps.hpp, 4262
- radix
  - std::\_\_numeric\_limits\_base, 1827
  - std::numeric\_limits< \_Tp >, 3449
- random, 4262, 4263
- Random Number Distributions, 315
- Random Number Generation, 316
  - generate\_canonical, 316
- Random Number Generators, 317
  - minstd\_rand, 318
  - minstd\_rand0, 318
  - mt19937, 318
  - mt19937\_64, 319
  - operator!=, 319–321
  - operator<=, 322
- Random Number Utilities, 323
- random.h, 4263
- random.tcc, 4267, 4271
- random\_number.h, 4273
- random\_sample
  - SGI, 381
- random\_sample\_n
  - SGI, 382
- random\_shuffle
  - Mutating, 194
- random\_shuffle.h, 4274
- random\_shuffle\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1313
- range\_access.h, 4274
- ranged\_hash\_fn.hpp, 4276
- ranged\_probe\_fn.hpp, 4276
- ratio, 4277, 4278
- ratio\_divide
  - Rational Arithmetic, 325
- ratio\_multiply
  - Rational Arithmetic, 325
- Rational Arithmetic, 324
  - ratio\_divide, 325
  - ratio\_multiply, 325
- rb\_tree, 4278
- rb\_tree.hpp, 4279
- rbegin

- `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 944
- `std`, 771, 772
- `std::basic_string< _CharT, _Traits, _Alloc >`, 2537
- `std::deque< _Tp, _Alloc >`, 2879
- `std::list< _Tp, _Alloc >`, 3178
- `std::map< _Key, _Tp, _Compare, _Alloc >`, 3246
- `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 3354, 3355
- `std::multiset< _Key, _Compare, _Alloc >`, 3383
- `std::set< _Key, _Compare, _Alloc >`, 3613
- `std::vector< _Tp, _Alloc >`, 3927
- `rc.hpp`, 4279
- `rc_binomial_heap.hpp`, 4280
- `rc_string_base.h`, 4280
- `rdbuf`
  - `std::basic_fstream< _CharT, _Traits >`, 2039, 2040
  - `std::basic_ifstream< _CharT, _Traits >`, 2096
  - `std::basic_ios< _CharT, _Traits >`, 2132
  - `std::basic_iostream< _CharT, _Traits >`, 2190, 2191
  - `std::basic_istream< _CharT, _Traits >`, 2245
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2300
  - `std::basic_ofstream< _CharT, _Traits >`, 2347, 2348
  - `std::basic_ostream< _CharT, _Traits >`, 2389, 2390
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 2434, 2435
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2620, 2621
- `rdstate`
  - `std::basic_fstream< _CharT, _Traits >`, 2040
  - `std::basic_ifstream< _CharT, _Traits >`, 2096
  - `std::basic_ios< _CharT, _Traits >`, 2133
  - `std::basic_iostream< _CharT, _Traits >`, 2191
  - `std::basic_istream< _CharT, _Traits >`, 2246
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2301
  - `std::basic_ofstream< _CharT, _Traits >`, 2348
  - `std::basic_ostream< _CharT, _Traits >`, 2390
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 2435
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2621
- `read`
  - `std::basic_fstream< _CharT, _Traits >`, 2040
  - `std::basic_ifstream< _CharT, _Traits >`, 2097
  - `std::basic_iostream< _CharT, _Traits >`, 2192
  - `std::basic_istream< _CharT, _Traits >`, 2246
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2301
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2621
- `readsome`
  - `std::basic_fstream< _CharT, _Traits >`, 2041
- `std::basic_ifstream< _CharT, _Traits >`, 2097
- `std::basic_iostream< _CharT, _Traits >`, 2192
- `std::basic_istream< _CharT, _Traits >`, 2247
- `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2302
- `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2622
- `ready`
  - `std::match_results< _Bi_iter, _Alloc >`, 3262
- `rebind`
  - `std::pointer_traits< _Ptr >`, 3523
- `ref`
  - `std`, 772
- `reference`
  - `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_iterator, Iterator, _Alloc >`, 1372
  - `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_iterator, Iterator, _Alloc >`, 1378
  - `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >`, 1387
  - `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`, 1392
  - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`, 1443
  - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`, 1449
  - `const_iterator_`, 1649
  - `iterator_`, 1654
  - `point_const_iterator_`, 1660
  - `point_iterator_`, 1665
  - `std::back_insert_iterator< _Container >`, 1956
  - `std::front_insert_iterator< _Container >`, 2972
  - `std::insert_iterator< _Container >`, 3051
  - `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`, 3135
  - `std::istreambuf_iterator< _CharT, _Traits >`, 3138
  - `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`, 3143
  - `std::ostream_iterator< _Tp, _CharT, _Traits >`, 3487
  - `std::ostreambuf_iterator< _CharT, _Traits >`, 3492
  - `std::raw_storage_iterator< _OutputIterator, _Tp >`, 3551
  - `std::set< _Key, _Compare, _Alloc >`, 3592
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3784
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3817
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3847
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3877
- `refwrap.h`, 4281
- `regex`, 4282
- Regular Expressions, 332



- regex.h, [4282](#)
- regex.tcc, [4287](#)
- regex\_automaton.h, [4288](#)
- regex\_automaton.tcc, [4289](#)
- regex\_compiler.h, [4289](#)
- regex\_compiler.tcc, [4290](#)
- regex\_constants.h, [4290](#)
- regex\_error
  - std::regex\_error, [3554](#)
- regex\_error.h, [4292](#)
- regex\_executor.h, [4293](#)
- regex\_executor.tcc, [4293](#)
- regex\_iterator
  - std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3556](#), [3557](#)
- regex\_match
  - Regular Expressions, [356–360](#)
- regex\_replace
  - Regular Expressions, [361](#), [362](#), [364](#), [365](#)
- regex\_scanner.h, [4293](#)
- regex\_scanner.tcc, [4294](#)
- regex\_search
  - Regular Expressions, [366–370](#)
- regex\_token\_iterator
  - std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3560–3562](#)
- regex\_traits
  - std::regex\_traits< \_Ch\_type >, [3565](#)
- register\_callback
  - std::basic\_fstream< \_CharT, \_Traits >, [2042](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2098](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2133](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2193](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2248](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2303](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2348](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2391](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2435](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2623](#)
  - std::ios\_base, [3065](#)
- Regular Expressions, [326](#)
  - cregex\_token\_iterator, [331](#)
  - csub\_match, [331](#)
  - operator!=, [333–337](#)
  - operator<, [337](#), [339–341](#)
  - operator<<, [341](#)
  - operator<=, [342–345](#)
  - operator>, [349–352](#)
  - operator>=, [352](#), [353](#), [355](#), [356](#)
  - operator==, [345–348](#)
  - regex, [332](#)
  - regex\_match, [356–360](#)
  - regex\_replace, [361](#), [362](#), [364](#), [365](#)
  - regex\_search, [366–370](#)
  - sregex\_token\_iterator, [332](#)
  - ssub\_match, [332](#)
  - swap, [371](#)
  - wcregex\_token\_iterator, [332](#)
  - wcsub\_match, [332](#)
  - wregex, [333](#)
  - wsregex\_token\_iterator, [333](#)
  - wssub\_match, [333](#)
- rehash
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3809](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3839](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3869](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3898](#)
- release
  - std::auto\_ptr< \_Tp >, [1952](#)
  - std::unique\_ptr< \_Tp, \_Dp >, [3769](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3776](#)
- remove
  - Mutating, [194](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2961](#)
  - std::list< \_Tp, \_Alloc >, [3178](#)
- remove\_all\_extents\_t
  - Metaprogramming, [179](#)
- remove\_const\_t
  - Metaprogramming, [180](#)
- remove\_copy
  - Mutating, [195](#)
- remove\_copy\_if
  - Mutating, [196](#)
- remove\_cv\_t
  - Metaprogramming, [180](#)
- remove\_extent\_t
  - Metaprogramming, [180](#)
- remove\_if
  - Mutating, [196](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2962](#)
  - std::list< \_Tp, \_Alloc >, [3179](#)
- remove\_pointer\_t
  - Metaprogramming, [180](#)
- remove\_reference\_t
  - Metaprogramming, [180](#)
- remove\_volatile\_t
  - Metaprogramming, [181](#)
- rend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [945](#)
  - std, [772](#), [773](#)

- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2537
- std::deque< \_Tp, \_Alloc >, 2880
- std::list< \_Tp, \_Alloc >, 3179
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3246
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3355
- std::multiset< \_Key, \_Compare, \_Alloc >, 3384
- std::set< \_Key, \_Compare, \_Alloc >, 3613
- std::vector< \_Tp, \_Alloc >, 3927
- replace
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 945–953
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1193–1197
  - Mutating, 197
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2537–2545
- replace\_copy
  - std, 774
- replace\_copy\_if
  - Mutating, 197
- replace\_if
  - Mutating, 198
- replace\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1313
- requested\_size
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 1097
  - std::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1891
- reserve
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 954
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1198
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2545
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3809
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3839
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3869
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3898
  - std::vector< \_Tp, \_Alloc >, 3928
- reset
  - std::auto\_ptr< \_Tp >, 1953
  - std::bernoulli\_distribution, 2643
  - std::binomial\_distribution< \_IntType >, 2655
  - std::bitset< \_Nb >, 2670
  - std::cauchy\_distribution< \_RealType >, 2676
  - std::chi\_squared\_distribution< \_RealType >, 2684
  - std::discrete\_distribution< \_IntType >, 2892
  - std::exponential\_distribution< \_RealType >, 2929
  - std::extreme\_value\_distribution< \_RealType >, 2935
  - std::fisher\_f\_distribution< \_RealType >, 2939
  - std::gamma\_distribution< \_RealType >, 2997
  - std::geometric\_distribution< \_IntType >, 3002
  - std::lognormal\_distribution< \_RealType >, 3213
  - std::negative\_binomial\_distribution< \_IntType >, 3393
  - std::normal\_distribution< \_RealType >, 3399
  - std::piecewise\_constant\_distribution< \_RealType >, 3509
  - std::piecewise\_linear\_distribution< \_RealType >, 3515
  - std::poisson\_distribution< \_IntType >, 3529
  - std::student\_t\_distribution< \_RealType >, 3652
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3729
  - std::uniform\_int\_distribution< \_IntType >, 3756
  - std::uniform\_real\_distribution< \_RealType >, 3761
  - std::unique\_ptr< \_Tp, \_Dp >, 3769
  - std::unique\_ptr< \_Tp[], \_Dp >, 3776
  - std::weibull\_distribution< \_RealType >, 3960
- resetiosflags
  - std, 774
- resize
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 954, 955
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 1561
  - Numeric Arrays, 280
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2546, 2547
  - std::deque< \_Tp, \_Alloc >, 2880, 2881
  - std::forward\_list< \_Tp, \_Alloc >, 2962
  - std::list< \_Tp, \_Alloc >, 3180
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3730
  - std::vector< \_Tp, \_Alloc >, 3928, 3929
- resize\_fn\_imps.hpp, 4294
- resize\_no\_store\_hash\_fn\_imps.hpp, 4294, 4295
- resize\_policy.hpp, 4295
- resize\_store\_hash\_fn\_imps.hpp, 4295
- result\_of\_t
  - Metaprogramming, 181
- result\_type
  - \_\_gnu\_cxx::\_\_detail::\_\_Ffit\_finder< \_Tp >, 878
  - \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >, 971
  - \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, 1023
  - \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, 1025
  - \_\_gnu\_cxx::select1st< \_Pair >, 1037
  - \_\_gnu\_cxx::select2nd< \_Pair >, 1039
  - \_\_gnu\_cxx::subtractive\_rng, 1094
  - \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >, 1108



- `__gnu_parallel::EqualFromLess< _T1, _T2, _Compare >`, 1250
- `__gnu_parallel::EqualTo< _T1, _T2 >`, 1252
- `__gnu_parallel::Less< _T1, _T2 >`, 1262
- `__gnu_parallel::Lexicographic< _T1, _T2, _Compare >`, 1263
- `__gnu_parallel::LexicographicReverse< _T1, _T2, _Compare >`, 1265
- `__gnu_parallel::Multiplies< _Tp1, _Tp2, _Result >`, 1288
- `__gnu_parallel::Plus< _Tp1, _Tp2, _Result >`, 1292
- `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 1207
- `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 1209
- `__gnu_parallel::__unary_negate< _Predicate, argument_type >`, 1244
- `std::_Maybe_unary_or_binary_function< _Res, _T1 >`, 1883
- `std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`, 1884
- `std::bernoulli_distribution`, 2641
- `std::binary_function< _Arg1, _Arg2, _Result >`, 2646
- `std::binary_negate< _Predicate >`, 2648
- `std::binder1st< _Operation >`, 2650
- `std::binder2nd< _Operation >`, 2651
- `std::binomial_distribution< _IntType >`, 2653
- `std::cauchy_distribution< _RealType >`, 2675
- `std::chi_squared_distribution< _RealType >`, 2682
- `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 2748
- `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 2750
- `std::const_mem_fun_ref_t< _Ret, _Tp >`, 2752
- `std::const_mem_fun_t< _Ret, _Tp >`, 2753
- `std::discard_block_engine< _RandomNumberEngine, __p, __r >`, 2883
- `std::discrete_distribution< _IntType >`, 2891
- `std::divides< _Tp >`, 2896
- `std::equal_to< _Tp >`, 2900
- `std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >`, 2923
- `std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >`, 2924
- `std::exponential_distribution< _RealType >`, 2927
- `std::extreme_value_distribution< _RealType >`, 2933
- `std::fisher_f_distribution< _RealType >`, 2937
- `std::gamma_distribution< _RealType >`, 2994
- `std::geometric_distribution< _IntType >`, 3000
- `std::greater< _Tp >`, 3005
- `std::greater_equal< _Tp >`, 3007
- `std::hash< __gnu_cxx::throw_value_limit >`, 3017
- `std::hash< __gnu_cxx::throw_value_random >`, 3018
- `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`, 3040
- `std::less< _Tp >`, 3147
- `std::less_equal< _Tp >`, 3149
- `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, 3151
- `std::logical_and< _Tp >`, 3205
- `std::logical_not< _Tp >`, 3207
- `std::logical_or< _Tp >`, 3209
- `std::lognormal_distribution< _RealType >`, 3212
- `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 3266
- `std::mem_fun1_t< _Ret, _Tp, _Arg >`, 3267
- `std::mem_fun_ref_t< _Ret, _Tp >`, 3269
- `std::mem_fun_t< _Ret, _Tp >`, 3270
- `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`, 3272
- `std::minus< _Tp >`, 3285
- `std::modulus< _Tp >`, 3287
- `std::multiplies< _Tp >`, 3360
- `std::negate< _Tp >`, 3388
- `std::negative_binomial_distribution< _IntType >`, 3391
- `std::normal_distribution< _RealType >`, 3397
- `std::not_equal_to< _Tp >`, 3403
- `std::owner_less< shared_ptr< _Tp > >`, 3498
- `std::owner_less< void >`, 3499
- `std::owner_less< weak_ptr< _Tp > >`, 3501
- `std::piecewise_constant_distribution< _RealType >`, 3507
- `std::piecewise_linear_distribution< _RealType >`, 3513
- `std::plus< _Tp >`, 3519
- `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`, 3520
- `std::pointer_to_unary_function< _Arg, _Result >`, 3522
- `std::poisson_distribution< _IntType >`, 3527
- `std::random_device`, 3544
- `std::seed_seq`, 3586
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 3638
- `std::student_t_distribution< _RealType >`, 3651
- `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, 3661
- `std::unary_function< _Arg, _Result >`, 3750
- `std::unary_negate< _Predicate >`, 3752
- `std::uniform_int_distribution< _IntType >`, 3754
- `std::uniform_real_distribution< _RealType >`, 3759
- `std::weibull_distribution< _RealType >`, 3958
- `rethrow_exception`
- `Exceptions`, 87
- `rethrow_if_nested`

- Exceptions, [87](#)
- return\_temporary\_buffer
  - std, [775](#)
- reverse
  - Mutating, [199](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2963](#)
  - std::list< \_Tp, \_Alloc >, [3180](#)
- reverse\_copy
  - Mutating, [199](#)
- reverse\_iteration
  - \_\_gnu\_pbds::container\_traits< Cntnr >, [1362](#)
- reverse\_iterator
  - std::reverse\_iterator< \_Iterator >, [3577](#), [3578](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3592](#)
- rfind
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [955–957](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1199](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [2547–2549](#)
- riemann\_zeta
  - Mathematical Special Functions, [158](#), [170](#)
- riemann\_zetaf
  - Mathematical Special Functions, [158](#)
- riemann\_zetal
  - Mathematical Special Functions, [159](#)
- right
  - std, [775](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2057](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2111](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2143](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2208](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2260](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2316](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2360](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2403](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2448](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2638](#)
  - std::ios\_base, [3074](#)
- rope, [4295](#)
- ropeimpl.h, [4299](#)
- rotate
  - Mutating, [200](#)
- rotate\_copy
  - Mutating, [200](#)
- rotate\_fn\_imps.hpp, [4299](#)
- round\_error
  - std::numeric\_limits< \_Tp >, [3445](#)
- round\_style
  - std::\_\_numeric\_limits\_base, [1827](#)
- std::numeric\_limits< \_Tp >, [3450](#)
- round\_to\_nearest
  - std, [673](#)
- round\_toward\_infinity
  - std, [673](#)
- round\_toward\_neg\_infinity
  - std, [673](#)
- round\_toward\_zero
  - std, [673](#)
- runtime\_error
  - std::runtime\_error, [3583](#)
- safe\_base.h, [4300](#)
- safe\_container.h, [4300](#)
- safe\_iterator.h, [4300](#)
- safe\_iterator.tcc, [4302](#)
- safe\_local\_iterator.h, [4303](#)
- safe\_local\_iterator.tcc, [4303](#)
- safe\_sequence.h, [4304](#)
- safe\_sequence.tcc, [4304](#)
- safe\_unordered\_base.h, [4305](#)
- safe\_unordered\_container.h, [4305](#)
- safe\_unordered\_container.tcc, [4305](#)
- sample
  - experimental/algorithm, [3981](#)
- sample\_probe\_fn
  - \_\_gnu\_pbds::sample\_probe\_fn, [1583](#)
- sample\_probe\_fn.hpp, [4306](#)
- sample\_range\_hashing
  - \_\_gnu\_pbds::sample\_range\_hashing, [1585](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1592](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1596](#)
  - \_\_gnu\_pbds::sample\_size\_policy, [1598](#)
- sample\_range\_hashing.hpp, [4306](#)
- sample\_ranged\_hash\_fn
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, [1587](#)
- sample\_ranged\_hash\_fn.hpp, [4306](#)
- sample\_ranged\_probe\_fn.hpp, [4307](#)
- sample\_resize\_policy
  - \_\_gnu\_pbds::sample\_resize\_policy, [1589](#)
- sample\_resize\_policy.hpp, [4307](#)
- sample\_resize\_trigger
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1594](#)
- sample\_resize\_trigger.hpp, [4307](#)
- sample\_size\_policy
  - \_\_gnu\_pbds::sample\_size\_policy, [1598](#)
- sample\_size\_policy.hpp, [4308](#)
- sample\_tree\_node\_update.hpp, [4308](#)
- sample\_trie\_access\_traits.hpp, [4308](#)
- sample\_trie\_node\_update
  - \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [1601](#)
- sample\_trie\_node\_update.hpp, [4309](#)
- sample\_update\_policy

- `__gnu_pbds::sample_update_policy`, 1602
- `sample_update_policy.hpp`, 4309
- `sputc`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 994
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 1058
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 1082
  - `std::basic_filebuf< _CharT, _Traits >`, 1979
  - `std::basic_streambuf< _CharT, _Traits >`, 2476
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2563
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3946
- `scan_is`
  - `std::__ctype_abstract_base< _CharT >`, 1692
  - `std::ctype< _CharT >`, 2766
  - `std::ctype< char >`, 2783
  - `std::ctype< wchar_t >`, 2801
  - `std::ctype_byname< _CharT >`, 2819
  - `std::ctype_byname< char >`, 2836
- `scan_not`
  - `std::__ctype_abstract_base< _CharT >`, 1693
  - `std::ctype< _CharT >`, 2766
  - `std::ctype< char >`, 2784
  - `std::ctype< wchar_t >`, 2802
  - `std::ctype_byname< _CharT >`, 2819
  - `std::ctype_byname< char >`, 2836
- `scientific`
  - `std`, 775
  - `std::basic_fstream< _CharT, _Traits >`, 2057
  - `std::basic_ifstream< _CharT, _Traits >`, 2112
  - `std::basic_ios< _CharT, _Traits >`, 2144
  - `std::basic_iostream< _CharT, _Traits >`, 2208
  - `std::basic_istream< _CharT, _Traits >`, 2261
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 2317
  - `std::basic_ofstream< _CharT, _Traits >`, 2361
  - `std::basic_ostream< _CharT, _Traits >`, 2403
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 2448
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2639
  - `std::ios_base`, 3074
- `scoped_allocator`, 4309
- `search`
  - Non-Mutating, 230, 231
- `search.h`, 4310
- `search_minimal_n`
  - `__gnu_parallel::Settings`, 1313
- `search_n`
  - Non-Mutating, 231, 232
- `second`
  - `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`, 1258
  - `std::pair< _T1, _T2 >`, 3505
  - `std::sub_match< _Biter >`, 3659
- `second_argument_type`
  - `__gnu_cxx::project1st< _Arg1, _Arg2 >`, 1023
  - `__gnu_cxx::project2nd< _Arg1, _Arg2 >`, 1025
  - `__gnu_parallel::EqualFromLess< _T1, _T2, _Compare >`, 1250
  - `__gnu_parallel::EqualTo< _T1, _T2 >`, 1252
  - `__gnu_parallel::Less< _T1, _T2 >`, 1262
  - `__gnu_parallel::Lexicographic< _T1, _T2, _Compare >`, 1263
  - `__gnu_parallel::LexicographicReverse< _T1, _T2, _Compare >`, 1265
  - `__gnu_parallel::Multiplies< _Tp1, _Tp2, _Result >`, 1288
  - `__gnu_parallel::Plus< _Tp1, _Tp2, _Result >`, 1292
  - `std::Maybe_unary_or_binary_function< _Res, _T1, _T2 >`, 1884
  - `std::binary_function< _Arg1, _Arg2, _Result >`, 2647
  - `std::binary_negate< _Predicate >`, 2648
  - `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 2748
  - `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 2750
  - `std::divides< _Tp >`, 2896
  - `std::equal_to< _Tp >`, 2900
  - `std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >`, 2923
  - `std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >`, 2924
  - `std::greater< _Tp >`, 3005
  - `std::greater_equal< _Tp >`, 3007
  - `std::less< _Tp >`, 3147
  - `std::less_equal< _Tp >`, 3149
  - `std::logical_and< _Tp >`, 3205
  - `std::logical_or< _Tp >`, 3209
  - `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 3266
  - `std::mem_fun1_t< _Ret, _Tp, _Arg >`, 3267
  - `std::minus< _Tp >`, 3285
  - `std::modulus< _Tp >`, 3287
  - `std::multiplies< _Tp >`, 3360
  - `std::not_equal_to< _Tp >`, 3403
  - `std::owner_less< shared_ptr< _Tp > >`, 3498
  - `std::owner_less< void >`, 3499
  - `std::owner_less< weak_ptr< _Tp > >`, 3501
  - `std::plus< _Tp >`, 3519
  - `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`, 3520
- `second_type`
  - `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`, 1257
  - `std::pair< _T1, _T2 >`, 3504
  - `std::sub_match< _Biter >`, 3656
- `seconds`
  - `std::chrono`, 828
- `seed`
  - `std::discard_block_engine< _RandomNumberEngine,`

- \_\_p, \_\_r >, [2887](#)
  - std::independent\_bits\_engine< \_\_RandomNumberEngine, \_\_w, \_UIntType >, [3043](#), [3044](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [3153](#), [3154](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [3641](#), [3642](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3662](#), [3663](#)
- seed\_seq
  - std::seed\_seq, [3586](#)
- seekdir
  - std::basic\_fstream< \_CharT, \_Traits >, [2002](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2068](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2121](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2155](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2218](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2273](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2326](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2370](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2414](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2584](#)
  - std::ios\_base, [3061](#)
- seekg
  - std::basic\_fstream< \_CharT, \_Traits >, [2042](#), [2043](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2099](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2194](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2248](#), [2249](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2303](#), [2304](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2623](#), [2624](#)
- seekoff
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [994](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1058](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1082](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1979](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2476](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2563](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3946](#)
- seekp
  - std::basic\_fstream< \_CharT, \_Traits >, [2043](#), [2044](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2195](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2349](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2391](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2436](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2624](#), [2625](#)
- seekpos
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [994](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1058](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1083](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1979](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2476](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2564](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3946](#)
- select\_on\_container\_copy\_construction
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [874](#)
  - std::allocator\_traits< \_Alloc >, [1915](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1922](#)
- sentry
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2263](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [2405](#)
- Sequences, [384](#)
- sequential
  - \_\_gnu\_parallel, [504](#)
- set, [4310](#), [4311](#)
  - \_\_gnu\_parallel::Settings, [1307](#)
  - std::bitset< \_Nb >, [2670](#), [2671](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3593](#)–[3597](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3730](#)
- Set Operation, [385](#)
  - includes, [386](#)
  - set\_difference, [387](#)
  - set\_intersection, [388](#), [389](#)
  - set\_symmetric\_difference, [390](#)
  - set\_union, [392](#), [393](#)
- set.h, [4312](#), [4313](#)
- set\_difference
  - Set Operation, [387](#)
- set\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1313](#)
- set\_intersection
  - Set Operation, [388](#), [389](#)
- set\_intersection\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1314](#)
- set\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1352](#)
- set\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [1556](#)
- set\_new\_handler
  - std, [776](#)
- set\_num\_threads
  - \_\_gnu\_parallel::balanced\_quicksort\_tag, [1318](#)
  - \_\_gnu\_parallel::balanced\_tag, [1319](#)
  - \_\_gnu\_parallel::default\_parallel\_tag, [1321](#)
  - \_\_gnu\_parallel::exact\_tag, [1323](#)
  - \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, [1326](#)

- `__gnu_parallel::multiway_mergesort_sampling_tag`, 1327
  - `__gnu_parallel::multiway_mergesort_tag`, 1329
  - `__gnu_parallel::omp_loop_static_tag`, 1330
  - `__gnu_parallel::omp_loop_tag`, 1332
  - `__gnu_parallel::parallel_tag`, 1334
  - `__gnu_parallel::quicksort_tag`, 1336
  - `__gnu_parallel::sampling_tag`, 1337
  - `__gnu_parallel::unbalanced_tag`, 1339
- `set_operations.h`, 4313
- `set_symmetric_difference`
  - Set Operation, 390
- `set_symmetric_difference_minimal_n`
  - `__gnu_parallel::Settings`, 1314
- `set_terminate`
  - std, 776
- `set_unexpected`
  - std, 776
- `set_union`
  - Set Operation, 392, 393
- `set_union_minimal_n`
  - `__gnu_parallel::Settings`, 1314
- `setbase`
  - std, 776
- `setbuf`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 995
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 1059
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 1083
  - `std::basic_filebuf<_CharT, _Traits>`, 1980
  - `std::basic_streambuf<_CharT, _Traits>`, 2477
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 2564
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3947
- `setf`
  - `std::basic_fstream<_CharT, _Traits>`, 2044, 2045
  - `std::basic_ifstream<_CharT, _Traits>`, 2100
  - `std::basic_ios<_CharT, _Traits>`, 2134
  - `std::basic_iostream<_CharT, _Traits>`, 2196
  - `std::basic_istream<_CharT, _Traits>`, 2249, 2250
  - `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, 2304, 2305
  - `std::basic_ofstream<_CharT, _Traits>`, 2350
  - `std::basic_ostream<_CharT, _Traits>`, 2392
  - `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, 2437
  - `std::basic_stringstream<_CharT, _Traits, _Alloc>`, 2625, 2626
  - `std::ios_base`, 3066
- `setfill`
  - std, 777
- `setg`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 995
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 1059
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 1084
  - `std::basic_filebuf<_CharT, _Traits>`, 1980
  - `std::basic_streambuf<_CharT, _Traits>`, 2477
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 2565
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3947
- `setiosflags`
  - std, 777
- `setp`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 996
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 1060
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 1084
  - `std::basic_filebuf<_CharT, _Traits>`, 1981
  - `std::basic_streambuf<_CharT, _Traits>`, 2478
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 2565
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3948
- `setprecision`
  - std, 777
- `setstate`
  - `std::basic_fstream<_CharT, _Traits>`, 2045
  - `std::basic_ifstream<_CharT, _Traits>`, 2101
  - `std::basic_ios<_CharT, _Traits>`, 2135
  - `std::basic_iostream<_CharT, _Traits>`, 2197
  - `std::basic_istream<_CharT, _Traits>`, 2250
  - `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, 2305
  - `std::basic_ofstream<_CharT, _Traits>`, 2351
  - `std::basic_ostream<_CharT, _Traits>`, 2393
  - `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, 2438
  - `std::basic_stringstream<_CharT, _Traits, _Alloc>`, 2626
- `settings.h`, 4314
  - `_GLIBCXX_PARALLEL_CONDITION`, 4315
- `setw`
  - std, 778
- `sgetc`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 996
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 1060
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 1085
  - `std::basic_filebuf<_CharT, _Traits>`, 1981
  - `std::basic_streambuf<_CharT, _Traits>`, 2478
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 2566
  - `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, 3948
- `sgetn`
  - `__gnu_cxx::enc_filebuf<_CharT>`, 997
  - `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`, 1061
  - `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, 1085
  - `std::basic_filebuf<_CharT, _Traits>`, 1982
  - `std::basic_streambuf<_CharT, _Traits>`, 2478
  - `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, 2566

- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3948
- SGL, 372
  - \_Find\_first, 375
  - \_Find\_next, 376
  - \_Unchecked\_flip, 376
  - \_Unchecked\_reset, 377
  - \_Unchecked\_set, 377
  - \_Unchecked\_test, 377
  - \_\_median, 374, 375
  - compose1, 377
  - compose2, 378
  - constant0, 378
  - constant1, 378
  - constant2, 378
  - copy\_n, 379
  - distance, 379
  - identity\_element, 379, 380
  - lexicographical\_compare\_3way, 380
  - power, 380, 381
  - random\_sample, 381
  - random\_sample\_n, 382
  - uninitialized\_copy\_n, 382
- shared\_future
  - std::shared\_future< \_Res >, 3619
  - std::shared\_future< \_Res & >, 3622
  - std::shared\_future< void >, 3625
- shared\_mutex, 4316
- shared\_ptr
  - std::shared\_ptr< \_Tp >, 3628–3635
- shared\_ptr.h, 4316, 4318
- shared\_ptr\_atomic.h, 4321
- shared\_ptr\_base.h, 4323
- shift
  - Numeric Arrays, 280
- showbase
  - std, 778
  - std::basic\_fstream< \_CharT, \_Traits >, 2058
  - std::basic\_ifstream< \_CharT, \_Traits >, 2112
  - std::basic\_ios< \_CharT, \_Traits >, 2144
  - std::basic\_iostream< \_CharT, \_Traits >, 2209
  - std::basic\_istream< \_CharT, \_Traits >, 2261
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2317
  - std::basic\_ofstream< \_CharT, \_Traits >, 2361
  - std::basic\_ostream< \_CharT, \_Traits >, 2403
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2449
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2639
  - std::ios\_base, 3074
- showmany
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 997
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1061
- \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 1086
- std::basic\_filebuf< \_CharT, \_Traits >, 1982
- std::basic\_streambuf< \_CharT, \_Traits >, 2479
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2567
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3949
- showpoint
  - std, 778
  - std::basic\_fstream< \_CharT, \_Traits >, 2058
  - std::basic\_ifstream< \_CharT, \_Traits >, 2112
  - std::basic\_ios< \_CharT, \_Traits >, 2144
  - std::basic\_iostream< \_CharT, \_Traits >, 2209
  - std::basic\_istream< \_CharT, \_Traits >, 2261
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2317
  - std::basic\_ofstream< \_CharT, \_Traits >, 2361
  - std::basic\_ostream< \_CharT, \_Traits >, 2403
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2449
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2639
  - std::ios\_base, 3074
- showpos
  - std, 778
  - std::basic\_fstream< \_CharT, \_Traits >, 2058
  - std::basic\_ifstream< \_CharT, \_Traits >, 2112
  - std::basic\_ios< \_CharT, \_Traits >, 2144
  - std::basic\_iostream< \_CharT, \_Traits >, 2209
  - std::basic\_istream< \_CharT, \_Traits >, 2261
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2317
  - std::basic\_ofstream< \_CharT, \_Traits >, 2361
  - std::basic\_ostream< \_CharT, \_Traits >, 2404
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2449
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2639
  - std::ios\_base, 3075
- shrink\_to\_fit
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 957
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 2549
  - std::deque< \_Tp, \_Alloc >, 2881
  - std::vector< \_Tp, \_Alloc >, 3929
- shuffle
  - Mutating, 201
- shuffle\_order\_engine
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3638–3640
- signaling\_NaN
  - std::numeric\_limits< \_Tp >, 3445
- sin
  - Complex Numbers, 68
- sinh



- Complex Numbers, [68](#)
- size
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [958](#)
  - `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`, [1098](#)
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1199](#)
  - Numeric Arrays, [281](#)
  - `std::Temporary_buffer< _ForwardIterator, _Tp >`, [1891](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [2549](#)
  - `std::bitset< _Nb >`, [2671](#)
  - `std::deque< _Tp, _Alloc >`, [2881](#)
  - `std::list< _Tp, _Alloc >`, [3181](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [3247](#)
  - `std::match_results< _Bi_iter, _Alloc >`, [3263](#)
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3355](#)
  - `std::multiset< _Key, _Compare, _Alloc >`, [3384](#)
  - `std::priority_queue< _Tp, _Sequence, _Compare >`, [3535](#)
  - `std::queue< _Tp, _Sequence >`, [3541](#)
  - `std::set< _Key, _Compare, _Alloc >`, [3614](#)
  - `std::stack< _Tp, _Sequence >`, [3648](#)
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, [3731](#)
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3810](#)
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3839](#)
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3870](#)
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3898](#)
  - `std::vector< _Tp, _Alloc >`, [3929](#)
- `size_fn_imps.hpp`, [4324](#)
- size\_type
  - `__gnu_pbds::hash_prime_size_policy`, [1557](#)
  - `__gnu_pbds::sample_range_hashing`, [1585](#)
  - `__gnu_pbds::sample_resize_policy`, [1589](#)
  - `__gnu_pbds::sample_resize_trigger`, [1593](#)
  - `__gnu_pbds::sample_size_policy`, [1597](#)
  - `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Ltr, _ATraits, _Alloc >`, [1622](#)
  - `std::allocator_traits< _Alloc >`, [1912](#)
  - `std::allocator_traits< allocator< _Tp > >`, [1919](#)
  - `std::set< _Key, _Compare, _Alloc >`, [3593](#)
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3784](#)
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3817](#)
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3847](#)
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3877](#)
- skipws
  - `std`, [779](#)
  - `std::basic_fstream< _CharT, _Traits >`, [2058](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [2112](#)
  - `std::basic_ios< _CharT, _Traits >`, [2144](#)
  - `std::basic_iostream< _CharT, _Traits >`, [2209](#)
  - `std::basic_istream< _CharT, _Traits >`, [2261](#)
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [2317](#)
  - `std::basic_ofstream< _CharT, _Traits >`, [2361](#)
  - `std::basic_ostream< _CharT, _Traits >`, [2404](#)
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, [2449](#)
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [2639](#)
  - `std::ios_base`, [3075](#)
  - `sleep_for`
    - `std::this_thread`, [858](#)
  - `sleep_until`
    - `std::this_thread`, [858](#)
  - slice
    - Numeric Arrays, [247](#)
  - `slice_array`
    - Numeric Arrays, [248](#)
  - `slice_array.h`, [4324](#)
  - `slist`, [4325](#)
  - `snextc`
    - `__gnu_cxx::enc_filebuf< _CharT >`, [998](#)
    - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [1062](#)
    - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, [1086](#)
    - `std::basic_filebuf< _CharT, _Traits >`, [1983](#)
    - `std::basic_streambuf< _CharT, _Traits >`, [2479](#)
    - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [2567](#)
    - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3949](#)
  - sort
    - Sorting, [414](#)
    - `std::forward_list< _Tp, _Alloc >`, [2963](#)
    - `std::list< _Tp, _Alloc >`, [3181](#)
  - `sort.h`, [4326](#)
  - sort\_heap
    - Heap, [108](#), [109](#)
  - sort\_minimal\_n
    - `__gnu_parallel::Settings`, [1314](#)
  - sort\_mwms\_oversampling
    - `__gnu_parallel::Settings`, [1314](#)
  - sort\_qs\_num\_samples\_preset
    - `__gnu_parallel::Settings`, [1315](#)
  - sort\_qsb\_base\_case\_maximal\_n
    - `__gnu_parallel::Settings`, [1315](#)
  - Sorting, [394](#)
    - `inplace_merge`, [396](#)
    - `is_sorted`, [397](#), [398](#)

- is\_sorted\_until, [398](#), [399](#)
- lexicographical\_compare, [399](#), [400](#)
- max, [400](#), [401](#)
- max\_element, [401](#), [402](#)
- merge, [402](#), [403](#)
- min, [404](#)
- min\_element, [405](#)
- minmax, [406](#)
- minmax\_element, [407](#)
- next\_permutation, [408](#)
- nth\_element, [409](#), [410](#)
- partial\_sort, [410](#), [411](#)
- partial\_sort\_copy, [411](#), [412](#)
- prev\_permutation, [413](#)
- sort, [414](#)
- stable\_sort, [415](#), [416](#)
- specfun.h, [4326](#)
- sph\_bessel
  - Mathematical Special Functions, [159](#), [170](#)
- sph\_besself
  - Mathematical Special Functions, [160](#)
- sph\_bessell
  - Mathematical Special Functions, [160](#)
- sph\_legendre
  - Mathematical Special Functions, [160](#), [170](#)
- sph\_legendref
  - Mathematical Special Functions, [161](#)
- sph\_legendrel
  - Mathematical Special Functions, [161](#)
- sph\_neumann
  - Mathematical Special Functions, [162](#), [171](#)
- sph\_neumannf
  - Mathematical Special Functions, [163](#)
- sph\_neumannl
  - Mathematical Special Functions, [163](#)
- splay\_fn\_imps.hpp, [4329](#)
- splay\_tree\_.hpp, [4329](#)
- splice
  - std::list< \_Tp, \_Alloc >, [3181](#)–[3183](#)
- splice\_after
  - std::forward\_list< \_Tp, \_Alloc >, [2964](#), [2965](#)
- split\_fn\_imps.hpp, [4330](#)
- split\_join\_can\_throw
  - \_\_gnu\_pbds::container\_traits< Cntnr >, [1362](#)
- split\_join\_fn\_imps.hpp, [4330](#), [4331](#)
- sputbackc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [998](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1062](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1087](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1983](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2480](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2567](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3950](#)
- sputc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [999](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1063](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1087](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1984](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2480](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2568](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3950](#)
- sputn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [999](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1063](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1088](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1984](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2481](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2568](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3951](#)
- sqrt
  - Complex Numbers, [68](#)
- sregex\_token\_iterator
  - Regular Expressions, [332](#)
- sso\_string\_base.h, [4331](#)
- sstream, [4331](#)
- sstream.tcc, [4332](#)
- ssub\_match
  - Regular Expressions, [332](#)
- stable\_partition
  - Mutating, [201](#)
- stable\_sort
  - Sorting, [415](#), [416](#)
- stack, [4333](#)
  - std::stack< \_Tp, \_Sequence >, [3647](#)
- standard\_policies.hpp, [4333](#)
- start
  - Numeric Arrays, [281](#), [282](#)
- state
  - std::fpos< \_StateT >, [2969](#), [2970](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3951](#)
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, [3965](#)
- static\_pointer\_cast
  - std, [779](#)
- std, [562](#)
  - \_Construct, [686](#)
  - \_Destroy, [686](#), [687](#)
  - \_Destroy\_n, [687](#)
  - \_allocate\_guarded, [673](#)
  - \_final\_insertion\_sort, [673](#)
  - \_find\_if, [674](#)
  - \_find\_if\_not, [674](#)
  - \_find\_if\_not\_n, [675](#)
  - \_gcd, [675](#)
  - \_gen\_two\_uniform\_ints, [675](#)



\_\_heap\_select, 676  
 \_\_inplace\_stable\_sort, 676  
 \_\_insertion\_sort, 676  
 \_\_introsort\_loop, 677  
 \_\_ioint, 788  
 \_\_lg, 677  
 \_\_merge\_adaptive, 677  
 \_\_merge\_without\_buffer, 678  
 \_\_move\_median\_to\_first, 678  
 \_\_move\_merge, 678  
 \_\_move\_merge\_adaptive, 679  
 \_\_move\_merge\_adaptive\_backward, 679  
 \_\_once\_call, 788  
 \_\_once\_callable, 789  
 \_\_once\_proxy, 679  
 \_\_partition, 680  
 \_\_ptr\_rebind, 669  
 \_\_reverse, 680, 681  
 \_\_rotate\_adaptive, 681  
 \_\_sample, 681, 682  
 \_\_search\_n\_aux, 682  
 \_\_stable\_partition\_adaptive, 683  
 \_\_try\_to\_lock, 683  
 \_\_umap\_traits, 669  
 \_\_ummap\_traits, 669  
 \_\_umset\_traits, 669  
 \_\_unguarded\_insertion\_sort, 684  
 \_\_unguarded\_linear\_insert, 684  
 \_\_unguarded\_partition, 684  
 \_\_unguarded\_partition\_pivot, 685  
 \_\_unique\_copy, 685, 686  
 \_\_uset\_traits, 669  
 accumulate, 687, 688  
 acos, 689  
 acosh, 689  
 adjacent\_difference, 689, 690  
 advance, 690  
 align, 691  
 arg, 692  
 asin, 692  
 asinh, 692  
 atan, 692  
 atanh, 692  
 begin, 693, 695  
 boolalpha, 695  
 call\_once, 695  
 cbegin, 695  
 cend, 696  
 cerr, 789  
 cin, 790  
 clog, 790  
 const\_pointer\_cast, 696  
 cout, 790  
 crbegin, 696  
 cref, 697  
 crend, 697  
 dec, 698  
 defaultfloat, 698  
 denorm\_absent, 673  
 denorm\_indeterminate, 673  
 denorm\_present, 673  
 distance, 698  
 dynamic\_pointer\_cast, 699  
 end, 699, 700  
 endl, 700  
 ends, 701  
 exchange, 701  
 fabs, 701  
 fixed, 701  
 float\_denorm\_style, 672  
 float\_round\_style, 673  
 flush, 702  
 get\_money, 702  
 get\_new\_handler, 702  
 get\_temporary\_buffer, 702  
 get\_terminate, 703  
 get\_time, 703  
 get\_unexpected, 704  
 getline, 704–706  
 hex, 707  
 hexfloat, 707  
 index\_sequence, 670  
 index\_sequence\_for, 670  
 inner\_product, 707, 708  
 internal, 708  
 io\_errc, 673  
 iota, 709  
 is\_nothrow\_swappable\_v, 790  
 is\_nothrow\_swappable\_with\_v, 790  
 is\_swappable\_v, 790  
 is\_swappable\_with\_v, 791  
 isalnum, 709  
 isalpha, 709  
 isblank, 710  
 iscntrl, 710  
 isdigit, 710  
 isgraph, 710  
 islower, 711  
 isprint, 711  
 ispunct, 711  
 isspace, 711  
 isupper, 712  
 isxdigit, 712  
 left, 712  
 lock, 712  
 make\_index\_sequence, 670  
 make\_integer\_sequence, 670  
 new\_handler, 670

- noboolalpha, 713
- noshowbase, 713
- noshowpoint, 713
- noshowpos, 714
- noskipws, 714
- nounitbuf, 714
- nouppercase, 714
- oct, 714
- operator!=, 715–719
- operator<, 722, 724–730
- operator<<, 730–738
- operator<=, 738–742
- operator>, 752–755
- operator>>, 760–766
- operator>=, 756–759
- operator^, 767
- operator+, 720–722
- operator==, 742–745, 747, 749–752
- operator&, 720
- operator|, 767
- partial\_sum, 768, 769
- put\_money, 769
- put\_time, 770
- quoted, 770
- rbegin, 771, 772
- ref, 772
- rend, 772, 773
- replace\_copy, 774
- resetiosflags, 774
- return\_temporary\_buffer, 775
- right, 775
- round\_to\_nearest, 673
- round\_toward\_infinity, 673
- round\_toward\_neg\_infinity, 673
- round\_toward\_zero, 673
- scientific, 775
- set\_new\_handler, 776
- set\_terminate, 776
- set\_unexpected, 776
- setbase, 776
- setfill, 777
- setiosflags, 777
- setprecision, 777
- setw, 778
- showbase, 778
- showpoint, 778
- showpos, 778
- skipws, 779
- static\_pointer\_cast, 779
- streamoff, 671
- streampos, 671
- streamsize, 671
- swap, 779–783
- terminate, 784
- terminate\_handler, 671
- tolower, 784
- toupper, 784
- try\_lock, 784
- u16streampos, 671
- u32streampos, 672
- uncaught\_exception, 785
- uncaught\_exceptions, 785
- unexpected, 785
- unexpected\_handler, 672
- uninitialized\_copy, 785
- uninitialized\_copy\_n, 786
- uninitialized\_fill, 786
- uninitialized\_fill\_n, 787
- unitbuf, 787
- uppercase, 788
- wcerr, 791
- wcin, 791
- wclog, 791
- wcout, 791
- ws, 788
- wstreampos, 672
- std::\_\_add\_pointer\_helper<\_Tp, bool>, 1667
- std::\_\_allocated\_ptr<\_Alloc>, 1668
  - \_\_allocated\_ptr, 1668, 1669
  - ~\_\_allocated\_ptr, 1669
  - get, 1669
  - operator=, 1669
- std::\_\_atomic\_base<\_ITp>, 1670
- std::\_\_atomic\_base<\_PTp\*>, 1671
- std::\_\_atomic\_flag\_base, 1672
- std::\_\_basic\_future<\_Res>, 1673
  - \_M\_get\_result, 1674
  - \_Ptr, 1674
- std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>, 1675
  - do\_out, 1677
  - in, 1677
  - out, 1678
  - unshift, 1679
- std::\_\_ctype\_abstract\_base<\_CharT>, 1680
  - char\_type, 1682
  - do\_is, 1682, 1683
  - do\_narrow, 1683, 1684
  - do\_scan\_is, 1685
  - do\_scan\_not, 1685
  - do\_tolower, 1686, 1687
  - do\_toupper, 1687, 1688
  - do\_widen, 1689
  - is, 1690
  - narrow, 1691, 1692
  - scan\_is, 1692
  - scan\_not, 1693
  - tolower, 1693, 1694

toupper, 1695  
 widen, 1696  
 std::\_\_debug, 792  
   operator<=, 796  
   operator>, 796  
   operator>=, 797  
   swap, 797  
 std::\_\_debug::bitset< \_Nb >, 1697  
 std::\_\_debug::deque< \_Tp, \_Allocator >, 1699  
   \_M\_const\_iterators, 1703  
   \_M\_detach\_all, 1701  
   \_M\_detach\_singular, 1701  
   \_M\_get\_mutex, 1701  
   \_M\_invalidate\_all, 1702  
   \_M\_invalidate\_if, 1702  
   \_M\_iterators, 1703  
   \_M\_revalidate\_singular, 1702  
   \_M\_swap, 1702  
   \_M\_transfer\_from\_if, 1702  
   \_M\_version, 1703  
 std::\_\_debug::forward\_list< \_Tp, \_Alloc >, 1704  
   \_M\_const\_iterators, 1708  
   \_M\_detach\_all, 1706  
   \_M\_detach\_singular, 1706  
   \_M\_get\_mutex, 1706  
   \_M\_invalidate\_all, 1707  
   \_M\_invalidate\_if, 1707  
   \_M\_iterators, 1708  
   \_M\_revalidate\_singular, 1707  
   \_M\_transfer\_from\_if, 1707  
   \_M\_version, 1708  
 std::\_\_debug::list< \_Tp, \_Allocator >, 1708  
   \_M\_const\_iterators, 1713  
   \_M\_detach\_all, 1711  
   \_M\_detach\_singular, 1711  
   \_M\_get\_mutex, 1711  
   \_M\_invalidate\_all, 1712  
   \_M\_invalidate\_if, 1712  
   \_M\_iterators, 1713  
   \_M\_revalidate\_singular, 1712  
   \_M\_swap, 1712  
   \_M\_transfer\_from\_if, 1712  
   \_M\_version, 1713  
 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >, 1714  
   \_M\_const\_iterators, 1718  
   \_M\_detach\_all, 1716  
   \_M\_detach\_singular, 1717  
   \_M\_get\_mutex, 1717  
   \_M\_invalidate\_all, 1717  
   \_M\_invalidate\_if, 1717  
   \_M\_iterators, 1718  
   \_M\_revalidate\_singular, 1718  
   \_M\_swap, 1718  
   \_M\_transfer\_from\_if, 1718  
   \_M\_version, 1719  
 std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, 1719  
   \_M\_const\_iterators, 1724  
   \_M\_detach\_all, 1722  
   \_M\_detach\_singular, 1722  
   \_M\_get\_mutex, 1722  
   \_M\_invalidate\_all, 1722  
   \_M\_invalidate\_if, 1723  
   \_M\_iterators, 1724  
   \_M\_revalidate\_singular, 1723  
   \_M\_swap, 1723  
   \_M\_transfer\_from\_if, 1723  
   \_M\_version, 1724  
 std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >, 1724  
   \_M\_const\_iterators, 1729  
   \_M\_detach\_all, 1727  
   \_M\_detach\_singular, 1727  
   \_M\_get\_mutex, 1727  
   \_M\_invalidate\_all, 1728  
   \_M\_invalidate\_if, 1728  
   \_M\_iterators, 1729  
   \_M\_revalidate\_singular, 1728  
   \_M\_swap, 1728  
   \_M\_transfer\_from\_if, 1728  
   \_M\_version, 1729  
 std::\_\_debug::set< \_Key, \_Compare, \_Allocator >, 1730  
   \_M\_const\_iterators, 1734  
   \_M\_detach\_all, 1732  
   \_M\_detach\_singular, 1732  
   \_M\_get\_mutex, 1733  
   \_M\_invalidate\_all, 1733  
   \_M\_invalidate\_if, 1733  
   \_M\_iterators, 1734  
   \_M\_revalidate\_singular, 1733  
   \_M\_swap, 1734  
   \_M\_transfer\_from\_if, 1734  
   \_M\_version, 1734  
 std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1735  
   \_M\_const\_iterators, 1740  
   \_M\_const\_local\_iterators, 1740  
   \_M\_detach\_all, 1738  
   \_M\_detach\_singular, 1738  
   \_M\_get\_mutex, 1738  
   \_M\_invalidate\_all, 1738  
   \_M\_invalidate\_if, 1739  
   \_M\_invalidate\_local\_if, 1739  
   \_M\_iterators, 1740  
   \_M\_local\_iterators, 1740  
   \_M\_revalidate\_singular, 1739  
   \_M\_swap, 1739

- [\\_M\\_version, 1741](#)
- [std::\\_\\_debug::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 1741](#)
- [\\_M\\_const\\_iterators, 1746](#)
- [\\_M\\_const\\_local\\_iterators, 1746](#)
- [\\_M\\_detach\\_all, 1744](#)
- [\\_M\\_detach\\_singular, 1744](#)
- [\\_M\\_get\\_mutex, 1744](#)
- [\\_M\\_invalidate\\_all, 1744](#)
- [\\_M\\_invalidate\\_if, 1745](#)
- [\\_M\\_invalidate\\_local\\_if, 1745](#)
- [\\_M\\_iterators, 1746](#)
- [\\_M\\_local\\_iterators, 1746](#)
- [\\_M\\_revalidate\\_singular, 1745](#)
- [\\_M\\_swap, 1745](#)
- [\\_M\\_version, 1747](#)
- [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 1747](#)
- [\\_M\\_const\\_iterators, 1752](#)
- [\\_M\\_const\\_local\\_iterators, 1752](#)
- [\\_M\\_detach\\_all, 1750](#)
- [\\_M\\_detach\\_singular, 1750](#)
- [\\_M\\_get\\_mutex, 1750](#)
- [\\_M\\_invalidate\\_all, 1750](#)
- [\\_M\\_invalidate\\_if, 1750](#)
- [\\_M\\_invalidate\\_local\\_if, 1751](#)
- [\\_M\\_iterators, 1752](#)
- [\\_M\\_local\\_iterators, 1752](#)
- [\\_M\\_revalidate\\_singular, 1751](#)
- [\\_M\\_swap, 1751](#)
- [\\_M\\_version, 1752](#)
- [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 1753](#)
- [\\_M\\_const\\_iterators, 1758](#)
- [\\_M\\_const\\_local\\_iterators, 1758](#)
- [\\_M\\_detach\\_all, 1756](#)
- [\\_M\\_detach\\_singular, 1756](#)
- [\\_M\\_get\\_mutex, 1756](#)
- [\\_M\\_invalidate\\_all, 1756](#)
- [\\_M\\_invalidate\\_if, 1756](#)
- [\\_M\\_invalidate\\_local\\_if, 1757](#)
- [\\_M\\_iterators, 1758](#)
- [\\_M\\_local\\_iterators, 1758](#)
- [\\_M\\_revalidate\\_singular, 1757](#)
- [\\_M\\_swap, 1757](#)
- [\\_M\\_version, 1758](#)
- [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 1759](#)
- [\\_M\\_const\\_iterators, 1764](#)
- [\\_M\\_detach\\_all, 1762](#)
- [\\_M\\_detach\\_singular, 1762](#)
- [\\_M\\_get\\_mutex, 1762](#)
- [\\_M\\_invalidate\\_all, 1762](#)
- [\\_M\\_invalidate\\_if, 1763](#)
- [\\_M\\_iterators, 1764](#)
- [\\_M\\_revalidate\\_singular, 1763](#)
- [\\_M\\_swap, 1763](#)
- [\\_M\\_transfer\\_from\\_if, 1763](#)
- [\\_M\\_version, 1764](#)
- [vector, 1762](#)
- [std::\\_\\_detail, 797](#)
- [operator<<, 800, 801](#)
- [operator>>, 801](#)
- [std::\\_\\_detail::BracketMatcher< \\_TraitsT, \\_\\_icase, \\_\\_colate >, 1764](#)
- [std::\\_\\_detail::Compiler< \\_TraitsT >, 1765](#)
- [std::\\_\\_detail::Default\\_ranged\\_hash, 1766](#)
- [std::\\_\\_detail::Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, \\_\\_cache\\_hash\\_code >, 1766](#)
- [std::\\_\\_detail::Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, false >, 1767](#)
- [std::\\_\\_detail::Equal\\_helper< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_HashCodeType, true >, 1767](#)
- [std::\\_\\_detail::Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, \\_Unique\\_keys >, 1768](#)
- [std::\\_\\_detail::Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >, 1769](#)
- [std::\\_\\_detail::Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >, 1770](#)
- [std::\\_\\_detail::Equality\\_base, 1770](#)
- [std::\\_\\_detail::Executor< \\_Bilter, \\_Alloc, \\_TraitsT, \\_\\_dfs\\_mode >, 1771](#)
- [std::\\_\\_detail::Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, false >, 1773](#)
- [std::\\_\\_detail::Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, true >, 1774](#)
- [std::\\_\\_detail::Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, \\_\\_cache\\_hash\\_code >, 1772](#)
- [std::\\_\\_detail::Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Hash, false >, 1776](#)
- [std::\\_\\_detail::Hash\\_node< \\_Value, \\_Cache\\_hash\\_code >, 1777](#)
- [std::\\_\\_detail::Hash\\_node< \\_Value, false >, 1777](#)
- [std::\\_\\_detail::Hash\\_node< \\_Value, true >, 1778](#)
- [std::\\_\\_detail::Hash\\_node\\_base, 1779](#)
- [std::\\_\\_detail::Hash\\_node\\_value\\_base< \\_Value >, 1780](#)
- [std::\\_\\_detail::Hashtable\\_alloc< \\_NodeAlloc >, 1781](#)
- [std::\\_\\_detail::Hashtable\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_Traits >, 1782](#)
- [std::\\_\\_detail::Hashtable\\_ebo\\_helper< \\_Nm, \\_Tp, \\_\\_use\\_ebo >, 1784](#)
- [std::\\_\\_detail::Hashtable\\_ebo\\_helper< \\_Nm, \\_Tp, false](#)

- [>, 1784](#)
- `std::__detail::Hashtable_ebo_helper< _Nm, _Tp, true >`, [1784](#)
- `std::__detail::Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >`, [1785](#)
- `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators >`, [1786](#)
- `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`, [1787](#)
- `std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`, [1788](#)
- `std::__detail::Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`, [1790](#)
- `std::__detail::List_node_base`, [1792](#)
- `std::__detail::List_node_header`, [1793](#)
- `std::__detail::Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`, [1794](#)
- `std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`, [1795](#)
- `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`, [1796](#)
- `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`, [1796](#)
- `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`, [1798](#)
- `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`, [1798](#)
- `std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`, [1797](#)
- `std::__detail::Mask_range_hashing`, [1799](#)
- `std::__detail::Mod_range_hashing`, [1800](#)
- `std::__detail::Node_const_iterator< _Value, __constant_iterators, __cache >`, [1800](#)
- `std::__detail::Node_iterator< _Value, __constant_iterators, __cache >`, [1802](#)
- `std::__detail::Node_iterator_base< _Value, _Cache_hash_code >`, [1803](#)
- `std::__detail::Power2_rehash_policy`, [1804](#)
- `std::__detail::Prime_rehash_policy`, [1804](#)
- `std::__detail::Quoted_string< _String, _CharT >`, [1805](#)
- `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::false_type >`, [1807](#)
- `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, std::true_type >`, [1807](#)
- `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename >`, [1806](#)
- `std::__detail::Scanner< _CharT >`, [1808](#)
- `std::__detail::TokenT`, [1809](#)
- `std::__detail::StateSeq< _TraitsT >`, [1810](#)
- `std::__detector< _Default, __void_t< _Op< _Args... > >, _Op, _Args... >`, [1811](#)
- `std::__detector< _Default, _AlwaysVoid, _Op, _Args >`, [1810](#)
- `std::__exception_ptr::exception_ptr`, [1811](#)
- `std::__future_base`, [1812](#)
- `std::__future_base::Ptr`, [1813](#)
- `std::__future_base::Result< _Res >`, [1814](#)
- `std::__future_base::Result< _Res & >`, [1815](#)
- `std::__future_base::Result< void >`, [1816](#)
- `std::__future_base::Result_alloc< _Res, _Alloc >`, [1817](#)
- `std::__future_base::Result_base`, [1818](#)
- `std::__is_location_invariant< _Tp >`, [1818](#)
- `std::__is_nullptr_t< _Tp >`, [1819](#)
- `std::__is_trivially_copy_assignable_impl< _Tp, bool >`, [1820](#)
- `std::__is_trivially_copy_constructible_impl< _Tp, bool >`, [1820](#)
- `std::__is_trivially_move_assignable_impl< _Tp, bool >`, [1820](#)
- `std::__is_trivially_move_constructible_impl< _Tp, bool >`, [1821](#)
- `std::__is_tuple_like_impl< std::pair< _T1, _T2 > >`, [1821](#)
- `std::__iterator_traits< _Iterator, typename >`, [1822](#)
- `std::__numeric_limits_base`, [1822](#)
- `digits`, [1823](#)
- `digits10`, [1824](#)
- `has_denorm`, [1824](#)
- `has_denorm_loss`, [1824](#)
- `has_infinity`, [1824](#)
- `has_quiet_NaN`, [1824](#)
- `has_signaling_NaN`, [1825](#)
- `is_bounded`, [1825](#)
- `is_exact`, [1825](#)
- `is_iec559`, [1825](#)
- `is_integer`, [1825](#)
- `is_modulo`, [1826](#)
- `is_signed`, [1826](#)
- `is_specialized`, [1826](#)
- `max_digits10`, [1826](#)
- `max_exponent`, [1826](#)
- `max_exponent10`, [1827](#)
- `min_exponent`, [1827](#)
- `min_exponent10`, [1827](#)
- `radix`, [1827](#)
- `round_style`, [1827](#)

tinyness\_before, 1828  
 traps, 1828  
 std::\_\_parallel, 801  
 std::\_\_parallel::\_\_CRandNumber< \_MustBeInt >, 1828  
 std::\_\_profile, 819  
   operator<=, 824  
   operator>, 824  
   operator>=, 824  
   swap, 824  
 std::\_\_profile::bitset< \_Nb >, 1829  
 std::\_\_profile::deque< \_Tp, \_Allocator >, 1830  
 std::\_\_profile::forward\_list< \_Tp, \_Alloc >, 1830  
 std::\_\_profile::list< \_Tp, \_Allocator >, 1832  
 std::\_\_profile::map< \_Key, \_Tp, \_Compare, \_Allocator >, 1834  
 std::\_\_profile::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, 1837  
 std::\_\_profile::multiset< \_Key, \_Compare, \_Allocator >, 1839  
 std::\_\_profile::set< \_Key, \_Compare, \_Allocator >, 1842  
 std::\_\_profile::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1845  
 std::\_\_profile::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 1847  
 std::\_\_profile::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 1848  
 std::\_\_profile::unordered\_set< \_Key, \_Hash, \_Pred, \_Alloc >, 1850  
 std::\_\_shared\_mutex\_cv, 1852  
 std::\_\_Base\_bitset< 0 >, 1854  
 std::\_\_Base\_bitset< 1 >, 1856  
 std::\_\_Base\_bitset< \_Nw >, 1853  
   \_M\_w, 1854  
 std::\_\_Bind< \_Signature >, 1857  
 std::\_\_Bind\_result< \_Result, \_Signature >, 1857  
 std::\_\_Deque\_base< \_Tp, \_Alloc >, 1858  
   \_M\_initialize\_map, 1859  
 std::\_\_Deque\_iterator< \_Tp, \_Ref, \_Ptr >, 1860  
   \_M\_set\_node, 1861  
 std::\_\_Enable\_copy\_move< \_Copy, \_CopyAssignment, \_Move, \_MoveAssignment, \_Tag >, 1862  
 std::\_\_Enable\_default\_constructor< \_Switch, \_Tag >, 1862  
 std::\_\_Enable\_destructor< \_Switch, \_Tag >, 1863  
 std::\_\_Enable\_special\_members< \_Default, \_Destructor, \_Copy, \_CopyAssignment, \_Move, \_MoveAssignment, \_Tag >, 1864  
 std::\_\_Function\_base, 1865  
 std::\_\_Fwd\_list\_base< \_Tp, \_Alloc >, 1866  
 std::\_\_Fwd\_list\_const\_iterator< \_Tp >, 1867  
 std::\_\_Fwd\_list\_iterator< \_Tp >, 1868  
 std::\_\_Fwd\_list\_node< \_Tp >, 1869  
 std::\_\_Fwd\_list\_node\_base, 1870  
 std::\_\_Hashtable< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >, 1871  
 std::\_\_List\_base< \_Tp, \_Alloc >, 1877  
 std::\_\_List\_const\_iterator< \_Tp >, 1878  
 std::\_\_List\_iterator< \_Tp >, 1879  
 std::\_\_List\_node< \_Tp >, 1880  
 std::\_\_Maybe\_get\_result\_type< \_Functor, typename >, 1881  
 std::\_\_Maybe\_unary\_or\_binary\_function< \_Res, \_ArgTypes >, 1882  
 std::\_\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1 >, 1882  
   argument\_type, 1883  
   result\_type, 1883  
 std::\_\_Maybe\_unary\_or\_binary\_function< \_Res, \_T1, \_T2 >, 1883  
   first\_argument\_type, 1884  
   result\_type, 1884  
   second\_argument\_type, 1884  
 std::\_\_Mu< \_Arg, \_IsBindExp, \_IsPlaceholder >, 1885  
 std::\_\_Mu< \_Arg, false, false >, 1885  
 std::\_\_Mu< \_Arg, false, true >, 1885  
 std::\_\_Mu< \_Arg, true, false >, 1886  
 std::\_\_Mu< reference\_wrapper< \_Tp >, false, false >, 1886  
 std::\_\_Not\_fn< \_Fn >, 1887  
 std::\_\_Placeholder< \_Num >, 1888  
 std::\_\_Reference\_wrapper\_base< \_Tp >, 1888  
 std::\_\_Sp\_ebo\_helper< \_Nm, \_Tp, false >, 1889  
 std::\_\_Sp\_ebo\_helper< \_Nm, \_Tp, true >, 1889  
 std::\_\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1890  
   \_Temporary\_buffer, 1891  
   begin, 1891  
   end, 1891  
   requested\_size, 1891  
   size, 1891  
 std::\_\_Tuple\_impl< \_Idx, \_Elements >, 1892  
 std::\_\_Tuple\_impl< \_Idx, \_Head, \_Tail... >, 1892  
 std::\_\_V2::condition\_variable\_any, 1894  
 std::\_\_V2::error\_category, 1895  
 std::\_\_Vector\_base< \_Tp, \_Alloc >, 1895  
 std::\_\_Weak\_result\_type< \_Functor >, 1896  
 std::\_\_Weak\_result\_type\_impl< \_Functor >, 1897  
 std::\_\_Weak\_result\_type\_impl< \_Res(\*)(\_ArgTypes...) \_GLIBCXX\_NOEXCEPT\_QUAL >, 1898  
 std::\_\_Weak\_result\_type\_impl< \_Res(\*)(\_ArgTypes.....) \_GLIBCXX\_NOEXCEPT\_QUAL >, 1898  
 std::\_\_Weak\_result\_type\_impl< \_Res(\_ArgTypes...) \_GLIBCXX\_NOEXCEPT\_QUAL >, 1898  
 std::\_\_Weak\_result\_type\_impl< \_Res(\_ArgTypes.....) \_GLIBCXX\_NOEXCEPT\_QUAL >, 1899  
 std::add\_const< \_Tp >, 1899  
 std::add\_cv< \_Tp >, 1900  
 std::add\_lvalue\_reference< \_Tp >, 1900  
 std::add\_rvalue\_reference< \_Tp >, 1901

std::add\_volatile< \_Tp >, 1901  
 std::adopt\_lock\_t, 1902  
 std::aligned\_storage< \_Len, \_Align >, 1902  
 std::aligned\_union< \_Len, \_Types >, 1903  
     type, 1903  
 std::alignment\_of< \_Tp >, 1904  
 std::allocator< \_Tp >, 1905  
 std::allocator< void >, 1907  
 std::allocator\_arg\_t, 1908  
 std::allocator\_traits< \_Alloc >, 1908  
     allocate, 1912, 1913  
     allocator\_type, 1909  
     const\_pointer, 1910  
     const\_void\_pointer, 1910  
     construct, 1913  
     deallocate, 1914  
     destroy, 1914  
     difference\_type, 1910  
     is\_always\_equal, 1910  
     max\_size, 1915  
     pointer, 1911  
     propagate\_on\_container\_copy\_assignment, 1911  
     propagate\_on\_container\_move\_assignment, 1911  
     propagate\_on\_container\_swap, 1911  
     select\_on\_container\_copy\_construction, 1915  
     size\_type, 1912  
     value\_type, 1912  
     void\_pointer, 1912  
 std::allocator\_traits< allocator< \_Tp > >, 1916  
     allocate, 1919, 1920  
     allocator\_type, 1917  
     const\_pointer, 1917  
     const\_void\_pointer, 1917  
     construct, 1920  
     deallocate, 1921  
     destroy, 1921  
     difference\_type, 1917  
     is\_always\_equal, 1918  
     max\_size, 1922  
     pointer, 1918  
     propagate\_on\_container\_copy\_assignment, 1918  
     propagate\_on\_container\_move\_assignment, 1918  
     propagate\_on\_container\_swap, 1918  
     select\_on\_container\_copy\_construction, 1922  
     size\_type, 1919  
     value\_type, 1919  
     void\_pointer, 1919  
 std::array< \_Tp, \_Nm >, 1923  
 std::atomic< \_Tp >, 1924  
 std::atomic< \_Tp \* >, 1925  
 std::atomic< bool >, 1927  
 std::atomic< char >, 1928  
 std::atomic< char16\_t >, 1929  
 std::atomic< char32\_t >, 1931  
 std::atomic< int >, 1932  
 std::atomic< long >, 1933  
 std::atomic< long long >, 1935  
 std::atomic< short >, 1936  
 std::atomic< signed char >, 1937  
 std::atomic< unsigned char >, 1939  
 std::atomic< unsigned int >, 1940  
 std::atomic< unsigned long >, 1941  
 std::atomic< unsigned long long >, 1943  
 std::atomic< unsigned short >, 1944  
 std::atomic< wchar\_t >, 1945  
 std::atomic\_flag, 1947  
 std::auto\_ptr< \_Tp >, 1947  
     ~auto\_ptr, 1950  
     auto\_ptr, 1949, 1950  
     element\_type, 1949  
     get, 1951  
     operator\*, 1951  
     operator->, 1951  
     operator=, 1951, 1952  
     release, 1952  
     reset, 1953  
 std::auto\_ptr\_ref< \_Tp1 >, 1953  
 std::back\_insert\_iterator< \_Container >, 1954  
     back\_insert\_iterator, 1956  
     container\_type, 1955  
     difference\_type, 1955  
     iterator\_category, 1955  
     operator\*, 1957  
     operator++, 1957  
     operator=, 1957  
     pointer, 1956  
     reference, 1956  
     value\_type, 1956  
 std::bad\_alloc, 1958  
     what, 1958  
 std::bad\_cast, 1959  
     what, 1960  
 std::bad\_exception, 1960  
     what, 1961  
 std::bad\_function\_call, 1961  
     what, 1962  
 std::bad\_typeid, 1962  
     what, 1963  
 std::bad\_weak\_ptr, 1963  
     what, 1964  
 std::basic\_filebuf< \_CharT, \_Traits >, 1964  
     \_M\_buf, 1988  
     \_M\_buf\_locale, 1988  
     \_M\_buf\_size, 1988  
     \_M\_create\_pback, 1968  
     \_M\_destroy\_pback, 1968  
     \_M\_ext\_buf, 1988  
     \_M\_ext\_buf\_size, 1988



[\\_M\\_ext\\_next, 1989](#)  
[\\_M\\_in\\_beg, 1989](#)  
[\\_M\\_in\\_cur, 1989](#)  
[\\_M\\_in\\_end, 1989](#)  
[\\_M\\_mode, 1989](#)  
[\\_M\\_out\\_beg, 1990](#)  
[\\_M\\_out\\_cur, 1990](#)  
[\\_M\\_out\\_end, 1990](#)  
[\\_M\\_pback, 1990](#)  
[\\_M\\_pback\\_cur\\_save, 1991](#)  
[\\_M\\_pback\\_end\\_save, 1991](#)  
[\\_M\\_pback\\_init, 1991](#)  
[\\_M\\_reading, 1992](#)  
[\\_M\\_set\\_buffer, 1968](#)  
[~basic\\_filebuf, 1968](#)  
[basic\\_filebuf, 1968](#)  
[close, 1969](#)  
[eback, 1969](#)  
[egptr, 1969](#)  
[epptr, 1970](#)  
[gbump, 1970](#)  
[getloc, 1971](#)  
[gptr, 1971](#)  
[imbue, 1971](#)  
[in\\_avail, 1972](#)  
[is\\_open, 1972](#)  
[open, 1972, 1973](#)  
[overflow, 1974](#)  
[pbackfail, 1974](#)  
[pbase, 1976](#)  
[pbump, 1976](#)  
[pptr, 1977](#)  
[pubimbue, 1977](#)  
[pubseekoff, 1977](#)  
[pubseekpos, 1978](#)  
[pubsetbuf, 1978](#)  
[pubsync, 1979](#)  
[sbumpc, 1979](#)  
[seekoff, 1979](#)  
[seekpos, 1979](#)  
[setbuf, 1980](#)  
[setg, 1980](#)  
[setp, 1981](#)  
[sgetc, 1981](#)  
[sgetn, 1982](#)  
[showmanyc, 1982](#)  
[snextc, 1983](#)  
[sputbackc, 1983](#)  
[sputc, 1984](#)  
[sputn, 1984](#)  
[sungetc, 1985](#)  
[sync, 1985](#)  
[uflow, 1985](#)  
[underflow, 1986](#)  
  
[xsgetn, 1986](#)  
[xspn, 1987](#)  
[std::basic\\_fstream<\\_CharT, \\_Traits >, 1993](#)  
[\\_M\\_gcount, 2052](#)  
[\\_M\\_getloc, 2004](#)  
[\\_M\\_write, 2004](#)  
[\\_\\_num\\_put\\_type, 2000](#)  
[~basic\\_fstream, 2004](#)  
[adjustfield, 2052](#)  
[app, 2052](#)  
[ate, 2052](#)  
[bad, 2005](#)  
[badbit, 2052](#)  
[basefield, 2053](#)  
[basic\\_fstream, 2003](#)  
[beg, 2053](#)  
[binary, 2053](#)  
[boolalpha, 2053](#)  
[clear, 2005](#)  
[close, 2005](#)  
[copyfmt, 2006](#)  
[cur, 2054](#)  
[dec, 2054](#)  
[end, 2054](#)  
[eof, 2006](#)  
[eofbit, 2054](#)  
[event, 2002](#)  
[event\\_callback, 2000](#)  
[exceptions, 2006, 2007](#)  
[fail, 2007](#)  
[failbit, 2055](#)  
[fill, 2008](#)  
[fixed, 2055](#)  
[flags, 2009](#)  
[floatfield, 2055](#)  
[flush, 2009](#)  
[fmtflags, 2000](#)  
[gcount, 2010](#)  
[get, 2010–2013](#)  
[getline, 2013–2015](#)  
[getloc, 2015](#)  
[good, 2015](#)  
[goodbit, 2055](#)  
[hex, 2056](#)  
[ignore, 2015, 2016](#)  
[imbue, 2017](#)  
[in, 2056](#)  
[init, 2017](#)  
[internal, 2056](#)  
[iostate, 2001](#)  
[is\\_open, 2018](#)  
[iword, 2018](#)  
[left, 2056](#)  
[narrow, 2018](#)



- oct, [2057](#)
- open, [2019](#)
- openmode, [2001](#)
- operator bool, [2020](#)
- operator!, [2020](#)
- operator<<, [2020–2028](#)
- operator>>, [2028–2036](#)
- out, [2057](#)
- peek, [2036](#)
- precision, [2037](#)
- put, [2038](#)
- putback, [2038](#)
- pword, [2039](#)
- rdbuf, [2039](#), [2040](#)
- rdstate, [2040](#)
- read, [2040](#)
- readsome, [2041](#)
- register\_callback, [2042](#)
- right, [2057](#)
- scientific, [2057](#)
- seekdir, [2002](#)
- seekg, [2042](#), [2043](#)
- seekp, [2043](#), [2044](#)
- setf, [2044](#), [2045](#)
- setstate, [2045](#)
- showbase, [2058](#)
- showpoint, [2058](#)
- showpos, [2058](#)
- skipws, [2058](#)
- sync, [2046](#)
- sync\_with\_stdio, [2046](#)
- tellg, [2047](#)
- tellp, [2047](#)
- tie, [2047](#), [2048](#)
- trunc, [2058](#)
- unget, [2048](#)
- unitbuf, [2059](#)
- unsetf, [2049](#)
- uppercase, [2059](#)
- widen, [2049](#)
- width, [2050](#)
- write, [2051](#)
- xalloc, [2051](#)
- std::basic\_ifstream< \_CharT, \_Traits >, [2060](#)
  - \_M\_gcount, [2106](#)
  - \_M\_getloc, [2070](#)
  - \_\_num\_put\_type, [2065](#)
  - ~basic\_ifstream, [2070](#)
  - adjustfield, [2106](#)
  - app, [2106](#)
  - ate, [2107](#)
  - bad, [2070](#)
  - badbit, [2107](#)
  - basefield, [2107](#)
  - basic\_ifstream, [2069](#)
  - beg, [2107](#)
  - binary, [2107](#)
  - boolalpha, [2108](#)
  - clear, [2071](#)
  - close, [2071](#)
  - copyfmt, [2071](#)
  - cur, [2108](#)
  - dec, [2108](#)
  - end, [2108](#)
  - eof, [2072](#)
  - eofbit, [2109](#)
  - event, [2068](#)
  - event\_callback, [2066](#)
  - exceptions, [2072](#)
  - fail, [2073](#)
  - failbit, [2109](#)
  - fill, [2073](#), [2074](#)
  - fixed, [2109](#)
  - flags, [2074](#), [2075](#)
  - floatfield, [2109](#)
  - fmtflags, [2066](#)
  - gcount, [2075](#)
  - get, [2075–2078](#)
  - getline, [2079](#), [2080](#)
  - getloc, [2080](#)
  - good, [2080](#)
  - goodbit, [2110](#)
  - hex, [2110](#)
  - ignore, [2081](#), [2082](#)
  - imbue, [2082](#)
  - in, [2110](#)
  - init, [2083](#)
  - internal, [2110](#)
  - iostate, [2067](#)
  - is\_open, [2083](#)
  - isword, [2083](#)
  - left, [2111](#)
  - narrow, [2084](#)
  - oct, [2111](#)
  - open, [2084](#), [2085](#)
  - openmode, [2067](#)
  - operator bool, [2085](#)
  - operator!, [2085](#)
  - operator>>, [2086–2093](#)
  - out, [2111](#)
  - peek, [2093](#)
  - precision, [2094](#)
  - putback, [2095](#)
  - pword, [2095](#)
  - rdbuf, [2096](#)
  - rdstate, [2096](#)
  - read, [2097](#)
  - readsome, [2097](#)

- register\_callback, 2098
- right, 2111
- scientific, 2112
- seekdir, 2068
- seekg, 2099
- setf, 2100
- setstate, 2101
- showbase, 2112
- showpoint, 2112
- showpos, 2112
- skipws, 2112
- sync, 2101
- sync\_with\_stdio, 2101
- tellg, 2102
- tie, 2102, 2103
- trunc, 2113
- unget, 2103
- unitbuf, 2113
- unsetf, 2104
- uppercase, 2113
- widen, 2104
- width, 2105
- xalloc, 2106
- std::basic\_ios< \_CharT, \_Traits >, 2114
  - \_M\_getloc, 2123
  - \_\_ctype\_type, 2117
  - \_\_num\_get\_type, 2117
  - \_\_num\_put\_type, 2118
  - ~basic\_ios, 2122
  - adjustfield, 2138
  - app, 2138
  - ate, 2139
  - bad, 2123
  - badbit, 2139
  - basefield, 2139
  - basic\_ios, 2122, 2123
  - beg, 2139
  - binary, 2139
  - boolalpha, 2140
  - char\_type, 2118
  - clear, 2124
  - copyfmt, 2124
  - cur, 2140
  - dec, 2140
  - end, 2140
  - eof, 2125
  - eofbit, 2141
  - event, 2122
  - event\_callback, 2118
  - exceptions, 2125
  - fail, 2126
  - failbit, 2141
  - fill, 2126
  - fixed, 2141
  - flags, 2127
  - floatfield, 2141
  - fmtflags, 2119
  - getloc, 2128
  - good, 2128
  - goodbit, 2142
  - hex, 2142
  - imbue, 2128
  - in, 2142
  - init, 2129
  - int\_type, 2119
  - internal, 2142
  - iostate, 2120
  - isword, 2129
  - left, 2143
  - narrow, 2130
  - oct, 2143
  - off\_type, 2120
  - openmode, 2120
  - operator bool, 2130
  - operator!, 2130
  - out, 2143
  - pos\_type, 2121
  - precision, 2131
  - pword, 2131
  - rdbuf, 2132
  - rdstate, 2133
  - register\_callback, 2133
  - right, 2143
  - scientific, 2144
  - seekdir, 2121
  - setf, 2134
  - setstate, 2135
  - showbase, 2144
  - showpoint, 2144
  - showpos, 2144
  - skipws, 2144
  - sync\_with\_stdio, 2135
  - tie, 2135, 2136
  - traits\_type, 2121
  - trunc, 2145
  - unitbuf, 2145
  - unsetf, 2136
  - uppercase, 2145
  - widen, 2137
  - width, 2137
  - xalloc, 2138
- std::basic\_istream< \_CharT, \_Traits >, 2146
  - \_M\_gcount, 2203
  - \_M\_getloc, 2156
  - \_M\_write, 2156
  - \_\_num\_put\_type, 2153
  - ~basic\_istream, 2156
  - adjustfield, 2203

app, 2203  
ate, 2203  
bad, 2157  
badbit, 2203  
basefield, 2204  
basic\_istream, 2156  
beg, 2204  
binary, 2204  
boolalpha, 2204  
clear, 2157  
copyfmt, 2157  
cur, 2205  
dec, 2205  
end, 2205  
eof, 2158  
eofbit, 2205  
event, 2155  
event\_callback, 2153  
exceptions, 2158  
fail, 2159  
failbit, 2206  
fill, 2159, 2160  
fixed, 2206  
flags, 2160, 2161  
floatfield, 2206  
flush, 2161  
fmtflags, 2153  
gcount, 2161  
get, 2162–2165  
getline, 2165, 2166  
getloc, 2167  
good, 2167  
goodbit, 2206  
hex, 2207  
ignore, 2167, 2168  
imbue, 2169  
in, 2207  
init, 2169  
internal, 2207  
iostate, 2154  
iword, 2170  
left, 2207  
narrow, 2170  
oct, 2208  
openmode, 2154  
operator bool, 2171  
operator!, 2171  
operator<<, 2171–2173, 2175–2179  
operator>>, 2180–2187  
out, 2208  
peek, 2188  
precision, 2188  
put, 2189  
putback, 2189

pword, 2190  
rdbuf, 2190, 2191  
rdstate, 2191  
read, 2192  
readsome, 2192  
register\_callback, 2193  
right, 2208  
scientific, 2208  
seekdir, 2155  
seekg, 2194  
seekp, 2195  
setf, 2196  
setstate, 2197  
showbase, 2209  
showpoint, 2209  
showpos, 2209  
skipws, 2209  
sync, 2197  
sync\_with\_stdio, 2197  
tellg, 2198  
tellp, 2198  
tie, 2198, 2199  
trunc, 2209  
unget, 2199  
unitbuf, 2210  
unsetf, 2200  
uppercase, 2210  
widen, 2200  
width, 2201  
write, 2202  
xalloc, 2202  
std::basic\_istream<\_CharT, \_Traits >, 2210  
    \_M\_gcount, 2255  
    \_M\_getloc, 2220  
    \_\_num\_put\_type, 2216  
    ~basic\_istream, 2219  
adjustfield, 2255  
app, 2256  
ate, 2256  
bad, 2220  
badbit, 2256  
basefield, 2256  
basic\_istream, 2219  
beg, 2256  
binary, 2257  
boolalpha, 2257  
clear, 2220  
copyfmt, 2221  
cur, 2257  
dec, 2257  
end, 2257  
eof, 2221  
eofbit, 2258  
event, 2219

- event\_callback, [2216](#)
- exceptions, [2221](#), [2222](#)
- fail, [2222](#)
- failbit, [2258](#)
- fill, [2223](#)
- fixed, [2258](#)
- flags, [2224](#)
- floatfield, [2258](#)
- fmtflags, [2217](#)
- gcount, [2224](#)
- get, [2225–2228](#)
- getline, [2228](#), [2229](#)
- getloc, [2230](#)
- good, [2230](#)
- goodbit, [2259](#)
- hex, [2259](#)
- ignore, [2230](#), [2231](#)
- imbue, [2232](#)
- in, [2259](#)
- init, [2232](#)
- internal, [2259](#)
- iostate, [2217](#)
- isword, [2233](#)
- left, [2260](#)
- narrow, [2233](#)
- oct, [2260](#)
- openmode, [2218](#)
- operator bool, [2234](#)
- operator!, [2234](#)
- operator>>, [2234–2236](#), [2238–2242](#)
- out, [2260](#)
- peek, [2243](#)
- precision, [2243](#)
- putback, [2244](#)
- pword, [2244](#)
- rdbuf, [2245](#)
- rdstate, [2246](#)
- read, [2246](#)
- readsome, [2247](#)
- register\_callback, [2248](#)
- right, [2260](#)
- scientific, [2261](#)
- seekdir, [2218](#)
- seekg, [2248](#), [2249](#)
- setf, [2249](#), [2250](#)
- setstate, [2250](#)
- showbase, [2261](#)
- showpoint, [2261](#)
- showpos, [2261](#)
- skipws, [2261](#)
- sync, [2251](#)
- sync\_with\_stdio, [2251](#)
- tellg, [2252](#)
- tie, [2252](#)
- trunc, [2262](#)
- unget, [2253](#)
- unitbuf, [2262](#)
- unsetf, [2253](#)
- uppercase, [2262](#)
- widen, [2254](#)
- width, [2254](#)
- xalloc, [2255](#)
- std::basic\_istream<\_CharT, \_Traits>::sentry, [2263](#)
- operator bool, [2264](#)
- sentry, [2263](#)
- traits\_type, [2263](#)
- std::basic\_istream<\_CharT, \_Traits, \_Alloc>, [2265](#)
- \_M\_gcount, [2311](#)
- \_M\_getloc, [2275](#)
- \_\_num\_put\_type, [2270](#)
- ~basic\_istream, [2274](#)
- adjustfield, [2311](#)
- app, [2312](#)
- ate, [2312](#)
- bad, [2275](#)
- badbit, [2312](#)
- basefield, [2312](#)
- basic\_istream, [2274](#)
- beg, [2312](#)
- binary, [2313](#)
- boolalpha, [2313](#)
- clear, [2275](#)
- copyfmt, [2276](#)
- cur, [2313](#)
- dec, [2313](#)
- end, [2313](#)
- eof, [2276](#)
- eofbit, [2314](#)
- event, [2273](#)
- event\_callback, [2271](#)
- exceptions, [2276](#), [2277](#)
- fail, [2277](#)
- failbit, [2314](#)
- fill, [2278](#)
- fixed, [2314](#)
- flags, [2279](#)
- floatfield, [2314](#)
- fmtflags, [2271](#)
- gcount, [2279](#)
- get, [2280–2283](#)
- getline, [2283](#), [2284](#)
- getloc, [2285](#)
- good, [2285](#)
- goodbit, [2315](#)
- hex, [2315](#)
- ignore, [2285](#), [2286](#)
- imbue, [2287](#)
- in, [2315](#)

init, [2287](#)  
internal, [2315](#)  
iostate, [2272](#)  
iword, [2288](#)  
left, [2316](#)  
narrow, [2288](#)  
oct, [2316](#)  
openmode, [2272](#)  
operator bool, [2289](#)  
operator!, [2289](#)  
operator>>, [2289–2291](#), [2293–2297](#)  
out, [2316](#)  
peek, [2298](#)  
precision, [2298](#)  
putback, [2299](#)  
pword, [2299](#)  
rdbuf, [2300](#)  
rdstate, [2301](#)  
read, [2301](#)  
readsome, [2302](#)  
register\_callback, [2303](#)  
right, [2316](#)  
scientific, [2317](#)  
seekdir, [2273](#)  
seekg, [2303](#), [2304](#)  
setf, [2304](#), [2305](#)  
setstate, [2305](#)  
showbase, [2317](#)  
showpoint, [2317](#)  
showpos, [2317](#)  
skipws, [2317](#)  
str, [2306](#)  
sync, [2306](#)  
sync\_with\_stdio, [2307](#)  
tellg, [2307](#)  
tie, [2308](#)  
trunc, [2318](#)  
unget, [2309](#)  
unitbuf, [2318](#)  
unsetf, [2309](#)  
uppercase, [2318](#)  
widen, [2310](#)  
width, [2310](#)  
xalloc, [2311](#)  
std::basic\_ofstream<\_CharT, \_Traits >, [2319](#)  
    \_M\_getloc, [2328](#)  
    \_M\_write, [2328](#)  
    \_\_num\_get\_type, [2324](#)  
    ~basic\_ofstream, [2328](#)  
adjustfield, [2355](#)  
app, [2356](#)  
ate, [2356](#)  
bad, [2329](#)  
badbit, [2356](#)  
basefield, [2356](#)  
basic\_ofstream, [2327](#)  
beg, [2356](#)  
binary, [2357](#)  
boolalpha, [2357](#)  
clear, [2329](#)  
close, [2329](#)  
copyfmt, [2330](#)  
cur, [2357](#)  
dec, [2357](#)  
end, [2357](#)  
eof, [2330](#)  
eofbit, [2358](#)  
event, [2326](#)  
event\_callback, [2324](#)  
exceptions, [2330](#), [2331](#)  
fail, [2331](#)  
failbit, [2358](#)  
fill, [2332](#)  
fixed, [2358](#)  
flags, [2333](#)  
floatfield, [2358](#)  
flush, [2333](#)  
fmtflags, [2324](#)  
getloc, [2334](#)  
good, [2334](#)  
goodbit, [2359](#)  
hex, [2359](#)  
imbue, [2334](#)  
in, [2359](#)  
init, [2335](#)  
internal, [2359](#)  
iostate, [2325](#)  
is\_open, [2335](#)  
iword, [2335](#)  
left, [2360](#)  
narrow, [2336](#)  
oct, [2360](#)  
open, [2336](#), [2337](#)  
openmode, [2325](#)  
operator bool, [2337](#)  
operator!, [2337](#)  
operator<<, [2338–2345](#)  
out, [2360](#)  
precision, [2345](#), [2346](#)  
put, [2346](#)  
pword, [2347](#)  
rdbuf, [2347](#), [2348](#)  
rdstate, [2348](#)  
register\_callback, [2348](#)  
right, [2360](#)  
scientific, [2361](#)  
seekdir, [2326](#)  
seekp, [2349](#)

- setf, [2350](#)
- setstate, [2351](#)
- showbase, [2361](#)
- showpoint, [2361](#)
- showpos, [2361](#)
- skipws, [2361](#)
- sync\_with\_stdio, [2351](#)
- tellp, [2351](#)
- tie, [2352](#)
- trunc, [2362](#)
- unitbuf, [2362](#)
- unsetf, [2352](#)
- uppercase, [2362](#)
- widen, [2353](#)
- width, [2353](#), [2354](#)
- write, [2354](#)
- xalloc, [2355](#)
- std::basic\_ostream< \_CharT, \_Traits >, [2363](#)
  - \_M\_getloc, [2371](#)
  - \_M\_write, [2371](#)
  - \_\_num\_get\_type, [2368](#)
  - ~basic\_ostream, [2371](#)
  - adjustfield, [2398](#)
  - app, [2398](#)
  - ate, [2398](#)
  - bad, [2372](#)
  - badbit, [2398](#)
  - basefield, [2398](#)
  - basic\_ostream, [2371](#)
  - beg, [2399](#)
  - binary, [2399](#)
  - boolalpha, [2399](#)
  - clear, [2372](#)
  - copyfmt, [2372](#)
  - cur, [2399](#)
  - dec, [2400](#)
  - end, [2400](#)
  - eof, [2373](#)
  - eofbit, [2400](#)
  - event, [2370](#)
  - event\_callback, [2368](#)
  - exceptions, [2373](#)
  - fail, [2374](#)
  - failbit, [2400](#)
  - fill, [2374](#), [2375](#)
  - fixed, [2401](#)
  - flags, [2375](#), [2376](#)
  - floatfield, [2401](#)
  - flush, [2376](#)
  - fmtflags, [2368](#)
  - getloc, [2376](#)
  - good, [2377](#)
  - goodbit, [2401](#)
  - hex, [2401](#)
  - imbue, [2377](#)
  - in, [2402](#)
  - init, [2378](#)
  - internal, [2402](#)
  - iostate, [2369](#)
  - isword, [2378](#)
  - left, [2402](#)
  - narrow, [2379](#)
  - oct, [2402](#)
  - openmode, [2369](#)
  - operator bool, [2379](#)
  - operator!, [2379](#)
  - operator<<, [2380–2387](#)
  - out, [2403](#)
  - precision, [2387](#), [2388](#)
  - put, [2388](#)
  - pword, [2389](#)
  - rdbuf, [2389](#), [2390](#)
  - rdstate, [2390](#)
  - register\_callback, [2391](#)
  - right, [2403](#)
  - scientific, [2403](#)
  - seekdir, [2370](#)
  - seekp, [2391](#)
  - setf, [2392](#)
  - setstate, [2393](#)
  - showbase, [2403](#)
  - showpoint, [2403](#)
  - showpos, [2404](#)
  - skipws, [2404](#)
  - sync\_with\_stdio, [2393](#)
  - tellp, [2394](#)
  - tie, [2394](#)
  - trunc, [2404](#)
  - unitbuf, [2404](#)
  - unsetf, [2395](#)
  - uppercase, [2404](#)
  - widen, [2395](#)
  - width, [2396](#)
  - write, [2396](#)
  - xalloc, [2397](#)
- std::basic\_ostream< \_CharT, \_Traits >::sentry, [2405](#)
  - ~sentry, [2406](#)
  - operator bool, [2406](#)
  - sentry, [2405](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2407](#)
  - \_M\_getloc, [2416](#)
  - \_M\_write, [2416](#)
  - \_\_num\_get\_type, [2412](#)
  - ~basic\_ostringstream, [2415](#)
  - adjustfield, [2443](#)
  - app, [2443](#)
  - ate, [2443](#)
  - bad, [2417](#)

badbit, [2443](#)  
basefield, [2444](#)  
basic\_ostringstream, [2415](#)  
beg, [2444](#)  
binary, [2444](#)  
boolalpha, [2444](#)  
clear, [2417](#)  
copyfmt, [2417](#)  
cur, [2445](#)  
dec, [2445](#)  
end, [2445](#)  
eof, [2418](#)  
eofbit, [2445](#)  
event, [2414](#)  
event\_callback, [2412](#)  
exceptions, [2418](#)  
fail, [2419](#)  
failbit, [2446](#)  
fill, [2419](#), [2420](#)  
fixed, [2446](#)  
flags, [2420](#), [2421](#)  
floatfield, [2446](#)  
flush, [2421](#)  
fmtflags, [2412](#)  
getloc, [2421](#)  
good, [2422](#)  
goodbit, [2446](#)  
hex, [2447](#)  
imbue, [2422](#)  
in, [2447](#)  
init, [2423](#)  
internal, [2447](#)  
iostate, [2413](#)  
iword, [2423](#)  
left, [2447](#)  
narrow, [2424](#)  
oct, [2448](#)  
openmode, [2413](#)  
operator bool, [2424](#)  
operator!, [2424](#)  
operator<<, [2425–2432](#)  
out, [2448](#)  
precision, [2432](#), [2433](#)  
put, [2433](#)  
pword, [2434](#)  
rdbuf, [2434](#), [2435](#)  
rdstate, [2435](#)  
register\_callback, [2435](#)  
right, [2448](#)  
scientific, [2448](#)  
seekdir, [2414](#)  
seekp, [2436](#)  
setf, [2437](#)  
setstate, [2438](#)  
showbase, [2449](#)  
showpoint, [2449](#)  
showpos, [2449](#)  
skipws, [2449](#)  
str, [2438](#)  
sync\_with\_stdio, [2439](#)  
tellp, [2439](#)  
tie, [2439](#), [2440](#)  
trunc, [2449](#)  
unitbuf, [2450](#)  
unsetf, [2440](#)  
uppercase, [2450](#)  
widen, [2441](#)  
width, [2441](#)  
write, [2442](#)  
xalloc, [2442](#)  
std::basic\_regex< \_Ch\_type, \_Rx\_traits >, [2450](#)  
    ~basic\_regex, [2456](#)  
    assign, [2456–2459](#)  
    basic\_regex, [2452–2455](#)  
    flags, [2459](#)  
    getloc, [2460](#)  
    imbue, [2460](#)  
    mark\_count, [2460](#)  
    operator=, [2460–2462](#)  
    swap, [2462](#)  
std::basic\_streambuf< \_CharT, \_Traits >, [2463](#)  
    \_M\_buf\_locale, [2484](#)  
    \_M\_in\_beg, [2485](#)  
    \_M\_in\_cur, [2485](#)  
    \_M\_in\_end, [2485](#)  
    \_M\_out\_beg, [2485](#)  
    \_M\_out\_cur, [2485](#)  
    \_M\_out\_end, [2486](#)  
    \_\_streambuf\_type, [2467](#)  
    ~basic\_streambuf, [2468](#)  
    basic\_streambuf, [2468](#)  
    char\_type, [2467](#)  
    eback, [2469](#)  
    egptr, [2469](#)  
    epptr, [2469](#)  
    gbump, [2470](#)  
    getloc, [2470](#)  
    gptr, [2470](#)  
    imbue, [2471](#)  
    in\_avail, [2471](#)  
    int\_type, [2467](#)  
    off\_type, [2467](#)  
    overflow, [2472](#)  
    pbackfail, [2472](#)  
    pbase, [2473](#)  
    pbump, [2473](#)  
    pos\_type, [2467](#)  
    pptr, [2474](#)

- pubimbue, [2474](#)
- pubseekoff, [2474](#)
- pubseekpos, [2475](#)
- pubsetbuf, [2475](#)
- pubsync, [2476](#)
- sbumpc, [2476](#)
- seekoff, [2476](#)
- seekpos, [2476](#)
- setbuf, [2477](#)
- setg, [2477](#)
- setp, [2478](#)
- sgetc, [2478](#)
- sgetn, [2478](#)
- showmanyc, [2479](#)
- snextc, [2479](#)
- sputbackc, [2480](#)
- sputc, [2480](#)
- sputn, [2481](#)
- sungetc, [2481](#)
- sync, [2482](#)
- traits\_type, [2468](#)
- uflow, [2482](#)
- underflow, [2482](#)
- xsggetn, [2483](#)
- xspn, [2484](#)
- std::basic\_string<\_CharT, \_Traits, \_Alloc >, [2486](#)
  - ~basic\_string, [2495](#)
  - append, [2495–2498](#)
  - assign, [2499–2503](#)
  - at, [2503](#), [2504](#)
  - back, [2504](#), [2505](#)
  - basic\_string, [2490–2495](#)
  - begin, [2505](#)
  - c\_str, [2505](#)
  - capacity, [2505](#)
  - cbegin, [2506](#)
  - cend, [2506](#)
  - clear, [2506](#)
  - compare, [2506–2509](#)
  - copy, [2510](#)
  - crbegin, [2511](#)
  - crend, [2511](#)
  - data, [2511](#)
  - empty, [2511](#)
  - end, [2512](#)
  - erase, [2512](#), [2513](#)
  - find, [2514](#), [2515](#)
  - find\_first\_not\_of, [2516–2518](#)
  - find\_first\_of, [2518–2520](#)
  - find\_last\_not\_of, [2520–2522](#)
  - find\_last\_of, [2522–2524](#)
  - front, [2524](#), [2525](#)
  - get\_allocator, [2525](#)
  - insert, [2525–2530](#)
  - length, [2531](#)
  - max\_size, [2531](#)
  - npos, [2551](#)
  - operator+=, [2531–2533](#)
  - operator=, [2533–2535](#)
  - operator[], [2535](#)
  - pop\_back, [2536](#)
  - push\_back, [2536](#)
  - rbegin, [2537](#)
  - rend, [2537](#)
  - replace, [2537–2545](#)
  - reserve, [2545](#)
  - resize, [2546](#), [2547](#)
  - rfind, [2547–2549](#)
  - shrink\_to\_fit, [2549](#)
  - size, [2549](#)
  - substr, [2550](#)
  - swap, [2550](#)
- std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc >, [2552](#)
  - \_M\_buf\_locale, [2573](#)
  - \_M\_in\_beg, [2573](#)
  - \_M\_in\_cur, [2573](#)
  - \_M\_in\_end, [2573](#)
  - \_M\_mode, [2573](#)
  - \_M\_out\_beg, [2574](#)
  - \_M\_out\_cur, [2574](#)
  - \_M\_out\_end, [2574](#)
  - basic\_stringbuf, [2554](#), [2555](#)
  - eback, [2555](#)
  - egptr, [2555](#)
  - eptr, [2556](#)
  - gbump, [2556](#)
  - getloc, [2557](#)
  - gptr, [2557](#)
  - imbue, [2557](#)
  - in\_avail, [2558](#)
  - overflow, [2558](#)
  - pbackfail, [2559](#)
  - pbase, [2559](#)
  - pbump, [2560](#)
  - pptr, [2560](#)
  - pubimbue, [2560](#)
  - pubseekoff, [2561](#)
  - pubseekpos, [2561](#)
  - pubsetbuf, [2563](#)
  - pubsync, [2563](#)
  - sbumpc, [2563](#)
  - seekoff, [2563](#)
  - seekpos, [2564](#)
  - setbuf, [2564](#)
  - setg, [2565](#)
  - setp, [2565](#)
  - sgetc, [2566](#)
  - sgetn, [2566](#)



- showmanyc, [2567](#)
- snextc, [2567](#)
- sputbackc, [2567](#)
- sputc, [2568](#)
- sputn, [2568](#)
- str, [2569](#)
- sungetc, [2570](#)
- sync, [2570](#)
- uflow, [2570](#)
- underflow, [2571](#)
- xsggetn, [2571](#)
- xsgputn, [2572](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2575](#)
  - \_M\_gcount, [2633](#)
  - \_M\_getloc, [2586](#)
  - \_M\_write, [2587](#)
  - \_\_num\_put\_type, [2582](#)
  - ~basic\_stringstream, [2586](#)
  - adjustfield, [2633](#)
  - app, [2633](#)
  - ate, [2634](#)
  - bad, [2587](#)
  - badbit, [2634](#)
  - basefield, [2634](#)
  - basic\_stringstream, [2585](#), [2586](#)
  - beg, [2634](#)
  - binary, [2634](#)
  - boolalpha, [2635](#)
  - clear, [2588](#)
  - copyfmt, [2588](#)
  - cur, [2635](#)
  - dec, [2635](#)
  - end, [2635](#)
  - eof, [2589](#)
  - eofbit, [2636](#)
  - event, [2585](#)
  - event\_callback, [2582](#)
  - exceptions, [2589](#)
  - fail, [2590](#)
  - failbit, [2636](#)
  - fill, [2590](#)
  - fixed, [2636](#)
  - flags, [2591](#)
  - floatfield, [2636](#)
  - flush, [2592](#)
  - fmtflags, [2583](#)
  - gcount, [2592](#)
  - get, [2592](#)–[2595](#)
  - getline, [2596](#), [2597](#)
  - getloc, [2597](#)
  - good, [2597](#)
  - goodbit, [2637](#)
  - hex, [2637](#)
  - ignore, [2598](#), [2599](#)
  - imbue, [2599](#)
  - in, [2637](#)
  - init, [2600](#)
  - internal, [2637](#)
  - iostate, [2583](#)
  - isword, [2600](#)
  - left, [2638](#)
  - narrow, [2601](#)
  - oct, [2638](#)
  - openmode, [2584](#)
  - operator bool, [2601](#)
  - operator!, [2601](#)
  - operator<<, [2602](#)–[2609](#)
  - operator>>, [2609](#)–[2617](#)
  - out, [2638](#)
  - peek, [2617](#)
  - precision, [2618](#)
  - put, [2619](#)
  - putback, [2619](#)
  - pword, [2620](#)
  - rdbuf, [2620](#), [2621](#)
  - rdstate, [2621](#)
  - read, [2621](#)
  - readsome, [2622](#)
  - register\_callback, [2623](#)
  - right, [2638](#)
  - scientific, [2639](#)
  - seekdir, [2584](#)
  - seekg, [2623](#), [2624](#)
  - seekp, [2624](#), [2625](#)
  - setf, [2625](#), [2626](#)
  - setstate, [2626](#)
  - showbase, [2639](#)
  - showpoint, [2639](#)
  - showpos, [2639](#)
  - skipws, [2639](#)
  - str, [2627](#)
  - sync, [2627](#)
  - sync\_with\_stdio, [2628](#)
  - tellg, [2628](#)
  - tellp, [2629](#)
  - tie, [2629](#)
  - trunc, [2640](#)
  - unget, [2630](#)
  - unitbuf, [2640](#)
  - unsetf, [2630](#)
  - uppercase, [2640](#)
  - widen, [2631](#)
  - width, [2631](#)
  - write, [2632](#)
  - xalloc, [2632](#)
- std::bernoulli\_distribution, [2640](#)
  - bernoulli\_distribution, [2642](#)
  - max, [2642](#)

- min, [2642](#)
- operator(), [2642](#)
- operator==, [2644](#)
- p, [2643](#)
- param, [2643](#)
- reset, [2643](#)
- result\_type, [2641](#)
- std::bernoulli\_distribution::param\_type, [2644](#)
- std::bidirectional\_iterator\_tag, [2645](#)
- std::binary\_function<\_Arg1, \_Arg2, \_Result >, [2646](#)
  - first\_argument\_type, [2646](#)
  - result\_type, [2646](#)
  - second\_argument\_type, [2647](#)
- std::binary\_negate<\_Predicate >, [2647](#)
  - first\_argument\_type, [2648](#)
  - result\_type, [2648](#)
  - second\_argument\_type, [2648](#)
- std::binder1st<\_Operation >, [2649](#)
  - argument\_type, [2650](#)
  - result\_type, [2650](#)
- std::binder2nd<\_Operation >, [2650](#)
  - argument\_type, [2651](#)
  - result\_type, [2651](#)
- std::binomial\_distribution<\_IntType >, [2652](#)
  - max, [2653](#)
  - min, [2653](#)
  - operator<<, [2656](#)
  - operator>>, [2656](#)
  - operator(), [2654](#)
  - operator==, [2656](#)
  - p, [2654](#)
  - param, [2654](#), [2655](#)
  - reset, [2655](#)
  - result\_type, [2653](#)
  - t, [2655](#)
- std::binomial\_distribution<\_IntType >::param\_type, [2657](#)
- std::bitset<\_Nb >, [2658](#)
  - all, [2664](#)
  - any, [2664](#)
  - bitset, [2662](#), [2663](#)
  - count, [2664](#)
  - flip, [2664](#), [2665](#)
  - none, [2665](#)
  - operator!=, [2665](#)
  - operator<<, [2666](#)
  - operator<=, [2666](#)
  - operator>>, [2667](#)
  - operator>=, [2667](#)
  - operator~, [2669](#)
  - operator^=, [2669](#)
  - operator==, [2667](#)
  - operator&=, [2666](#)
  - operator[], [2668](#)
  - operator|=, [2669](#)
  - reset, [2670](#)
  - set, [2670](#), [2671](#)
  - size, [2671](#)
  - test, [2671](#)
  - to\_string, [2672](#)
  - to\_ulong, [2672](#)
- std::bitset<\_Nb >::reference, [2673](#)
- std::cauchy\_distribution<\_RealType >, [2674](#)
  - max, [2675](#)
  - min, [2675](#)
  - operator(), [2675](#)
  - operator==, [2676](#)
  - param, [2675](#), [2676](#)
  - reset, [2676](#)
  - result\_type, [2675](#)
- std::cauchy\_distribution<\_RealType >::param\_type, [2677](#)
- std::char\_traits<\_\_gnu\_cxx::character<\_Value, \_Int, \_St >>, [2679](#)
- std::char\_traits<\_CharT >, [2677](#)
- std::char\_traits<char >, [2679](#)
- std::char\_traits<wchar\_t >, [2680](#)
- std::chi\_squared\_distribution<\_RealType >, [2681](#)
  - max, [2683](#)
  - min, [2683](#)
  - operator<<, [2684](#)
  - operator>>, [2685](#)
  - operator(), [2683](#)
  - operator==, [2685](#)
  - param, [2683](#), [2684](#)
  - reset, [2684](#)
  - result\_type, [2682](#)
- std::chi\_squared\_distribution<\_RealType >::param\_type, [2686](#)
- std::chrono, [825](#)
  - duration\_cast, [828](#)
  - hours, [827](#)
  - microseconds, [827](#)
  - milliseconds, [827](#)
  - minutes, [827](#)
  - nanoseconds, [828](#)
  - seconds, [828](#)
  - time\_point\_cast, [828](#)
- std::chrono::\_V2::steady\_clock, [2686](#)
- std::chrono::\_V2::system\_clock, [2687](#)
- std::chrono::duration<\_Rep, \_Period >, [2688](#)
- std::chrono::duration\_values<\_Rep >, [2689](#)
- std::chrono::time\_point<\_Clock, \_Dur >, [2689](#)
- std::chrono::treat\_as\_floating\_point<\_Rep >, [2690](#)
- std::codecvt<\_InternT, \_ExternT, \_StateT >, [2691](#)
  - do\_out, [2692](#)
  - in, [2693](#)
  - out, [2694](#)
  - unshift, [2695](#)

std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2696  
     do\_out, 2698  
     in, 2698  
     out, 2699  
     unshift, 2700  
 std::codecvt< char, char, mbstate\_t >, 2701  
     do\_out, 2702  
     in, 2703  
     out, 2704  
     unshift, 2705  
 std::codecvt< char16\_t, char, mbstate\_t >, 2706  
     do\_out, 2707  
     in, 2708  
     out, 2709  
     unshift, 2710  
 std::codecvt< char32\_t, char, mbstate\_t >, 2711  
     do\_out, 2712  
     in, 2713  
     out, 2714  
     unshift, 2715  
 std::codecvt< wchar\_t, char, mbstate\_t >, 2716  
     do\_out, 2717  
     in, 2718  
     out, 2719  
     unshift, 2720  
 std::codecvt\_base, 2721  
 std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2722  
     do\_out, 2723  
     in, 2724  
     out, 2725  
     unshift, 2726  
 std::collate< \_CharT >, 2727  
     ~collate, 2730  
     char\_type, 2729  
     collate, 2729, 2730  
     compare, 2730  
     do\_compare, 2731  
     do\_hash, 2731  
     do\_transform, 2732  
     hash, 2733  
     id, 2734  
     string\_type, 2729  
     transform, 2733  
 std::collate\_byname< \_CharT >, 2734  
     char\_type, 2736  
     compare, 2736  
     do\_compare, 2737  
     do\_hash, 2737  
     do\_transform, 2738  
     hash, 2739  
     id, 2740  
     string\_type, 2736  
     transform, 2739  
 std::common\_type< \_Tp >, 2740  
 std::complex< \_Tp >, 2741  
     complex, 2742  
     operator+=", 2742  
     operator-=", 2743  
     value\_type, 2742  
 std::complex< double >, 2743  
 std::complex< float >, 2744  
 std::complex< long double >, 2745  
 std::condition\_variable, 2746  
 std::conditional< \_Cond, \_Iftrue, \_Iffalse >, 2747  
 std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 2747  
     first\_argument\_type, 2748  
     result\_type, 2748  
     second\_argument\_type, 2748  
 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 2749  
     first\_argument\_type, 2750  
     result\_type, 2750  
     second\_argument\_type, 2750  
 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 2751  
     argument\_type, 2751  
     result\_type, 2752  
 std::const\_mem\_fun\_t< \_Ret, \_Tp >, 2752  
     argument\_type, 2753  
     result\_type, 2753  
 std::ctype< \_CharT >, 2754  
     do\_is, 2756, 2757  
     do\_narrow, 2757, 2758  
     do\_scan\_is, 2758  
     do\_scan\_not, 2759  
     do\_tolower, 2760  
     do\_toupper, 2761  
     do\_widen, 2762, 2763  
     id, 2771  
     is, 2763, 2764  
     narrow, 2764, 2765  
     scan\_is, 2766  
     scan\_not, 2766  
     tolower, 2767  
     toupper, 2769  
     widen, 2770  
 std::ctype< char >, 2772  
     ~ctype, 2775  
     char\_type, 2774  
     classic\_table, 2775  
     ctype, 2774, 2775  
     do\_narrow, 2775, 2776  
     do\_tolower, 2777  
     do\_toupper, 2778  
     do\_widen, 2779  
     id, 2788  
     is, 2780  
     narrow, 2782  
     scan\_is, 2783

- scan\_not, [2784](#)
- table, [2784](#)
- table\_size, [2788](#)
- tolower, [2785](#)
- toupper, [2786](#)
- widen, [2787](#), [2788](#)
- std::ctype< wchar\_t >, [2789](#)
  - ~ctype, [2792](#)
  - char\_type, [2791](#)
  - ctype, [2791](#), [2792](#)
  - do\_is, [2792](#), [2793](#)
  - do\_narrow, [2793](#), [2794](#)
  - do\_scan\_is, [2795](#)
  - do\_scan\_not, [2795](#)
  - do\_tolower, [2796](#)
  - do\_toupper, [2797](#)
  - do\_widen, [2798](#), [2799](#)
  - id, [2806](#)
  - is, [2799](#), [2800](#)
  - narrow, [2800](#), [2801](#)
  - scan\_is, [2801](#)
  - scan\_not, [2802](#)
  - tolower, [2803](#)
  - toupper, [2804](#)
  - widen, [2805](#)
- std::ctype\_base, [2806](#)
- std::ctype\_byname< \_CharT >, [2807](#)
  - do\_is, [2809](#), [2810](#)
  - do\_narrow, [2810](#), [2811](#)
  - do\_scan\_is, [2811](#)
  - do\_scan\_not, [2812](#)
  - do\_tolower, [2813](#)
  - do\_toupper, [2814](#)
  - do\_widen, [2815](#), [2816](#)
  - id, [2824](#)
  - is, [2816](#), [2817](#)
  - narrow, [2817](#), [2818](#)
  - scan\_is, [2819](#)
  - scan\_not, [2819](#)
  - tolower, [2820](#)
  - toupper, [2822](#)
  - widen, [2823](#)
- std::ctype\_byname< char >, [2825](#)
  - char\_type, [2827](#)
  - classic\_table, [2827](#)
  - do\_narrow, [2827](#), [2829](#)
  - do\_tolower, [2829](#), [2830](#)
  - do\_toupper, [2830](#), [2831](#)
  - do\_widen, [2831](#), [2832](#)
  - id, [2841](#)
  - is, [2833](#)
  - narrow, [2834](#)
  - scan\_is, [2836](#)
  - scan\_not, [2836](#)
  - table, [2837](#)
  - table\_size, [2841](#)
  - tolower, [2837](#), [2838](#)
  - toupper, [2838](#), [2839](#)
  - widen, [2840](#)
- std::decay< \_Tp >, [2841](#)
- std::decimal, [829](#)
  - decimal32\_to\_long\_long, [838](#)
- std::decimal::decimal128, [2842](#)
  - decimal128, [2843](#)
- std::decimal::decimal32, [2844](#)
  - decimal32, [2845](#)
- std::decimal::decimal64, [2845](#)
  - decimal64, [2847](#)
- std::default\_delete< \_Tp >, [2847](#)
  - default\_delete, [2848](#)
  - operator(), [2848](#)
- std::default\_delete< \_Tp[] >, [2849](#)
  - default\_delete, [2849](#)
  - operator(), [2850](#)
- std::defer\_lock\_t, [2850](#)
- std::deque< \_Tp, \_Alloc >, [2850](#)
  - \_M\_fill\_initialize, [2860](#)
  - \_M\_impl, [2882](#)
  - \_M\_initialize\_map, [2861](#)
  - \_M\_new\_elements\_at\_back, [2862](#)
  - \_M\_new\_elements\_at\_front, [2862](#)
  - \_M\_pop\_back\_aux, [2862](#)
  - \_M\_pop\_front\_aux, [2862](#)
  - \_M\_push\_back\_aux, [2862](#)
  - \_M\_push\_front\_aux, [2863](#)
  - \_M\_range\_check, [2863](#)
  - \_M\_range\_initialize, [2863](#), [2864](#)
  - \_M\_reallocate\_map, [2864](#)
  - \_M\_reserve\_elements\_at\_back, [2865](#)
  - \_M\_reserve\_elements\_at\_front, [2865](#)
  - \_M\_reserve\_map\_at\_back, [2865](#)
  - \_M\_reserve\_map\_at\_front, [2865](#)
  - ~deque, [2860](#)
  - assign, [2866](#), [2867](#)
  - at, [2867](#), [2868](#)
  - back, [2868](#), [2869](#)
  - begin, [2869](#)
  - cbegin, [2869](#)
  - cend, [2869](#)
  - clear, [2870](#)
  - crbegin, [2870](#)
  - crend, [2870](#)
  - deque, [2856](#)–[2859](#)
  - emplace, [2870](#)
  - empty, [2871](#)
  - end, [2871](#)
  - erase, [2871](#), [2872](#)
  - front, [2872](#)

[get\\_allocator](#), 2873  
[insert](#), 2873–2875  
[max\\_size](#), 2876  
[operator=](#), 2876, 2877  
[operator\[\]](#), 2877, 2878  
[pop\\_back](#), 2878  
[pop\\_front](#), 2878  
[push\\_back](#), 2878  
[push\\_front](#), 2879  
[rbegin](#), 2879  
[rend](#), 2880  
[resize](#), 2880, 2881  
[shrink\\_to\\_fit](#), 2881  
[size](#), 2881  
[swap](#), 2881  
[std::discard\\_block\\_engine< \\_RandomNumberEngine, \\_\\_p, \\_\\_r >](#), 2882  
[base](#), 2885  
[discard](#), 2886  
[discard\\_block\\_engine](#), 2884, 2885  
[max](#), 2886  
[min](#), 2886  
[operator<=](#), 2888  
[operator>](#), 2888  
[operator\(\)](#), 2886  
[operator==](#), 2888  
[result\\_type](#), 2883  
[seed](#), 2887  
[std::discrete\\_distribution< \\_IntType >](#), 2889  
[max](#), 2891  
[min](#), 2891  
[operator<=](#), 2893  
[operator>](#), 2893  
[operator\(\)](#), 2891  
[operator==](#), 2893  
[param](#), 2891, 2892  
[probabilities](#), 2892  
[reset](#), 2892  
[result\\_type](#), 2891  
[std::discrete\\_distribution< \\_IntType >::param\\_type](#), 2894  
[std::divides< \\_Tp >](#), 2895  
[first\\_argument\\_type](#), 2895  
[result\\_type](#), 2896  
[second\\_argument\\_type](#), 2896  
[std::divides< void >](#), 2896  
[std::domain\\_error](#), 2897  
[what](#), 2898  
[std::enable\\_if< bool, \\_Tp >](#), 2898  
[std::enable\\_shared\\_from\\_this< \\_Tp >](#), 2898  
[std::equal\\_to< \\_Tp >](#), 2899  
[first\\_argument\\_type](#), 2900  
[result\\_type](#), 2900  
[second\\_argument\\_type](#), 2900  
[std::equal\\_to< void >](#), 2901  
[std::error\\_code](#), 2901  
[std::error\\_condition](#), 2902  
[std::exception](#), 2903  
[std::experimental::filesystem::v1::path](#), 2904  
[std::experimental::filesystem::v1::path::iterator](#), 2906  
[std::experimental::fundamentals\\_v1::\\_Has\\_addressof< \\_Tp >](#), 2907  
[std::experimental::fundamentals\\_v1::\\_Optional\\_base< \\_Tp, \\_ShouldProvideDestructor >](#), 2908  
[std::experimental::fundamentals\\_v1::\\_Optional\\_base< \\_Tp, false >](#), 2909  
[std::experimental::fundamentals\\_v1::any](#), 2910  
[~any](#), 2912  
[any](#), 2910, 2911  
[clear](#), 2912  
[empty](#), 2912  
[operator=](#), 2912, 2913  
[swap](#), 2913  
[type](#), 2913  
[std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#), 2914  
[what](#), 2915  
[std::experimental::fundamentals\\_v1::bad\\_optional\\_access](#), 2915  
[what](#), 2916  
[std::experimental::fundamentals\\_v1::basic\\_string\\_view< \\_CharT, \\_Traits >](#), 2916  
[std::experimental::fundamentals\\_v1::in\\_place\\_t](#), 2918  
[std::experimental::fundamentals\\_v1::nullopt\\_t](#), 2919  
[std::experimental::fundamentals\\_v1::optional< \\_Tp >](#), 2919  
[std::experimental::fundamentals\\_v2::ostream\\_joiner< \\_DelimT, \\_CharT, \\_Traits >](#), 2921  
[std::experimental::fundamentals\\_v2::owner\\_less< shared\\_ptr< \\_Tp > >](#), 2922  
[first\\_argument\\_type](#), 2923  
[result\\_type](#), 2923  
[second\\_argument\\_type](#), 2923  
[std::experimental::fundamentals\\_v2::owner\\_less< weak\\_ptr< \\_Tp > >](#), 2923  
[first\\_argument\\_type](#), 2924  
[result\\_type](#), 2924  
[second\\_argument\\_type](#), 2924  
[std::experimental::fundamentals\\_v2::propagate\\_const< \\_Tp >](#), 2925  
[std::exponential\\_distribution< \\_RealType >](#), 2926  
[exponential\\_distribution](#), 2927  
[lambda](#), 2928  
[max](#), 2928  
[min](#), 2928  
[operator\(\)](#), 2928  
[operator==](#), 2929  
[param](#), 2928, 2929  
[reset](#), 2929  
[result\\_type](#), 2927

`std::exponential_distribution< _RealType >::param_type`,  
2930

`std::extent< typename, _Uint >`, 2931

`std::extreme_value_distribution< _RealType >`, 2932

- `a`, 2933
- `b`, 2933
- `max`, 2933
- `min`, 2933
- `operator()`, 2934
- `operator==`, 2935
- `param`, 2934
- `reset`, 2935
- `result_type`, 2933

`std::extreme_value_distribution< _RealType >::param_type`, `std::front_insert_iterator< _Container >`, 2970  
2935

`std::fisher_f_distribution< _RealType >`, 2936

- `max`, 2937
- `min`, 2938
- `operator<<`, 2939
- `operator>>`, 2940
- `operator()`, 2938
- `operator==`, 2939
- `param`, 2938
- `reset`, 2939
- `result_type`, 2937

`std::fisher_f_distribution< _RealType >::param_type`,  
2940

`std::forward_iterator_tag`, 2941

`std::forward_list< _Tp, _Alloc >`, 2942

- `~forward_list`, 2949
- `assign`, 2949, 2950
- `before_begin`, 2950, 2951
- `begin`, 2951
- `cbegin`, 2952
- `cend`, 2952
- `clear`, 2952
- `emplace_after`, 2952
- `emplace_front`, 2953
- `empty`, 2953
- `end`, 2953, 2954
- `erase_after`, 2954
- `forward_list`, 2945–2948
- `front`, 2955
- `get_allocator`, 2955
- `insert_after`, 2956, 2957
- `max_size`, 2958
- `merge`, 2958, 2959
- `operator=`, 2959, 2960
- `pop_front`, 2960
- `push_front`, 2961
- `remove`, 2961
- `remove_if`, 2962
- `resize`, 2962
- `reverse`, 2963
- `sort`, 2963
- `splice_after`, 2964, 2965
- `swap`, 2966
- `unique`, 2966

`std::fpos< _StateT >`, 2967

- `fpos`, 2968
- `operator streamoff`, 2968
- `operator+`, 2968
- `operator+=`, 2968
- `operator-`, 2969
- `operator-=`, 2969
- `state`, 2969, 2970

`std::front_insert_iterator< _Container >`, 2970

- `container_type`, 2971
- `difference_type`, 2971
- `front_insert_iterator`, 2973
- `iterator_category`, 2972
- `operator*`, 2973
- `operator++`, 2973
- `operator=`, 2973
- `pointer`, 2972
- `reference`, 2972
- `value_type`, 2972

`std::function< _Res(_ArgTypes...)>`, 2974

- `function`, 2976, 2977
- `operator bool`, 2978
- `operator()`, 2978
- `operator=`, 2979–2981
- `swap`, 2981
- `target`, 2982
- `target_type`, 2982

`std::future< _Res >`, 2983

- `_M_get_result`, 2985
- `_Ptr`, 2985
- `future`, 2985
- `get`, 2985

`std::future< _Res & >`, 2986

- `_M_get_result`, 2988
- `_Ptr`, 2987
- `future`, 2988
- `get`, 2988

`std::future< void >`, 2989

- `_M_get_result`, 2991
- `_Ptr`, 2990
- `future`, 2991
- `get`, 2991

`std::future_error`, 2992

- `what`, 2992

`std::gamma_distribution< _RealType >`, 2993

- `alpha`, 2995
- `beta`, 2995
- `gamma_distribution`, 2994
- `max`, 2995

- min, 2995
- operator<<, 2997
- operator>>, 2998
- operator(), 2995, 2996
- operator==, 2997
- param, 2996
- reset, 2997
- result\_type, 2994
- std::gamma\_distribution< \_RealType >::param\_type, 2998
- std::geometric\_distribution< \_IntType >, 2999
  - max, 3001
  - min, 3001
  - operator(), 3001
  - operator==, 3002
  - p, 3001
  - param, 3001, 3002
  - reset, 3002
  - result\_type, 3000
- std::geometric\_distribution< \_IntType >::param\_type, 3003
- std::greater< \_Tp >, 3004
  - first\_argument\_type, 3004
  - result\_type, 3005
  - second\_argument\_type, 3005
- std::greater< void >, 3005
- std::greater\_equal< \_Tp >, 3006
  - first\_argument\_type, 3007
  - result\_type, 3007
  - second\_argument\_type, 3007
- std::greater\_equal< void >, 3008
- std::gslice, 3008
- std::gslice\_array< \_Tp >, 3009
- std::has\_virtual\_destructor< \_Tp >, 3011
- std::hash< \_\_debug::bitset< \_Nb > >, 3012
- std::hash< \_\_debug::vector< bool, \_Alloc > >, 3013
- std::hash< \_\_gnu\_cxx::\_\_u16vstring >, 3013
- std::hash< \_\_gnu\_cxx::\_\_u32vstring >, 3014
- std::hash< \_\_gnu\_cxx::\_\_vstring >, 3014
- std::hash< \_\_gnu\_cxx::\_\_wvstring >, 3015
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 3016
  - argument\_type, 3016
  - result\_type, 3017
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, 3017
  - argument\_type, 3018
  - result\_type, 3018
- std::hash< \_\_profile::bitset< \_Nb > >, 3018
- std::hash< \_\_profile::vector< bool, \_Alloc > >, 3019
- std::hash< \_\_shared\_ptr< \_Tp, \_Lp > >, 3020
- std::hash< \_Tp >, 3012
- std::hash< \_Tp \* >, 3020
- std::hash< bool >, 3021
- std::hash< char >, 3021
- std::hash< char16\_t >, 3022
- std::hash< char32\_t >, 3023
- std::hash< double >, 3023
- std::hash< error\_code >, 3024
- std::hash< experimental::shared\_ptr< \_Tp > >, 3024
- std::hash< float >, 3025
- std::hash< int >, 3026
- std::hash< long >, 3026
- std::hash< long double >, 3027
- std::hash< long long >, 3027
- std::hash< shared\_ptr< \_Tp > >, 3028
- std::hash< short >, 3029
- std::hash< signed char >, 3029
- std::hash< string >, 3030
- std::hash< thread::id >, 3030
- std::hash< type\_index >, 3031
- std::hash< u16string >, 3032
- std::hash< u32string >, 3032
- std::hash< unique\_ptr< \_Tp, \_Dp > >, 3033
- std::hash< unsigned char >, 3034
- std::hash< unsigned int >, 3034
- std::hash< unsigned long >, 3035
- std::hash< unsigned long long >, 3035
- std::hash< unsigned short >, 3036
- std::hash< wchar\_t >, 3037
- std::hash< wstring >, 3037
- std::hash<::bitset< \_Nb > >, 3038
- std::hash<::vector< bool, \_Alloc > >, 3038
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 3039
  - base, 3042
  - discard, 3042
  - independent\_bits\_engine, 3040–3042
  - max, 3042
  - min, 3043
  - operator>>, 3045
  - operator(), 3043
  - operator==, 3044
  - result\_type, 3040
  - seed, 3043, 3044
- std::indirect\_array< \_Tp >, 3045
- std::initializer\_list< \_E >, 3047
- std::input\_iterator\_tag, 3048
- std::insert\_iterator< \_Container >, 3049
  - container\_type, 3050
  - difference\_type, 3050
  - insert\_iterator, 3051
  - iterator\_category, 3050
  - operator\*, 3051
  - operator++, 3052
  - operator=, 3052
  - pointer, 3050
  - reference, 3051
  - value\_type, 3051
- std::integer\_sequence< \_Tp, \_Idx >, 3053



`std::integral_constant< _Tp, __v >`, 3054  
`std::invalid_argument`, 3055  
    what, 3056  
`std::ios_base`, 3056  
    \_M\_getloc, 3062  
    ~ios\_base, 3062  
    adjustfield, 3069  
    app, 3069  
    ate, 3069  
    badbit, 3069  
    basefield, 3069  
    beg, 3070  
    binary, 3070  
    boolalpha, 3070  
    cur, 3070  
    dec, 3071  
    end, 3071  
    eofbit, 3071  
    event, 3061  
    event\_callback, 3059  
    failbit, 3071  
    fixed, 3072  
    flags, 3062  
    floatfield, 3072  
    fmtflags, 3059  
    getloc, 3063  
    goodbit, 3072  
    hex, 3072  
    imbue, 3063  
    in, 3073  
    internal, 3073  
    iostate, 3060  
    iword, 3064  
    left, 3073  
    oct, 3073  
    openmode, 3060  
    out, 3074  
    precision, 3064  
    pword, 3065  
    register\_callback, 3065  
    right, 3074  
    scientific, 3074  
    seekdir, 3061  
    setf, 3066  
    showbase, 3074  
    showpoint, 3074  
    showpos, 3075  
    skipws, 3075  
    sync\_with\_stdio, 3067  
    trunc, 3075  
    unitbuf, 3075  
    unsetf, 3067  
    uppercase, 3075  
    width, 3067, 3068  
    xalloc, 3068  
`std::ios_base::failure`, 3076  
    what, 3077  
`std::is_abstract< _Tp >`, 3077  
`std::is_arithmetic< _Tp >`, 3078  
`std::is_array< typename >`, 3078  
`std::is_assignable< _Tp, _Up >`, 3079  
`std::is_base_of< _Base, _Derived >`, 3080  
`std::is_bind_expression< _Bind< _Signature > >`, 3082  
`std::is_bind_expression< _Bind_result< _Result, _Signature > >`, 3083  
`std::is_bind_expression< _Tp >`, 3081  
`std::is_bind_expression< const _Bind< _Signature > >`, 3084  
`std::is_bind_expression< const _Bind_result< _Result, _Signature > >`, 3085  
`std::is_bind_expression< const volatile _Bind< _Signature > >`, 3086  
`std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`, 3087  
`std::is_bind_expression< volatile _Bind< _Signature > >`, 3088  
`std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`, 3089  
`std::is_class< _Tp >`, 3090  
`std::is_compound< _Tp >`, 3091  
`std::is_const< typename >`, 3092  
`std::is_constructible< _Tp, _Args >`, 3093  
`std::is_convertible< _From, _To >`, 3094  
`std::is_copy_assignable< _Tp >`, 3095  
`std::is_copy_constructible< _Tp >`, 3095  
`std::is_default_constructible< _Tp >`, 3096  
`std::is_destructible< _Tp >`, 3097  
`std::is_empty< _Tp >`, 3097  
`std::is_enum< _Tp >`, 3098  
`std::is_error_code_enum< _Tp >`, 3099  
`std::is_error_code_enum< future_errc >`, 3100  
`std::is_error_condition_enum< _Tp >`, 3101  
`std::is_final< _Tp >`, 3102  
`std::is_floating_point< _Tp >`, 3103  
`std::is_function< typename >`, 3103  
`std::is_fundamental< _Tp >`, 3104  
`std::is_integral< _Tp >`, 3105  
`std::is_literal_type< _Tp >`, 3106  
`std::is_lvalue_reference< typename >`, 3107  
`std::is_member_function_pointer< _Tp >`, 3108  
`std::is_member_object_pointer< _Tp >`, 3109  
`std::is_member_pointer< _Tp >`, 3109  
`std::is_move_assignable< _Tp >`, 3110  
`std::is_move_constructible< _Tp >`, 3110  
`std::is_nothrow_assignable< _Tp, _Up >`, 3111  
`std::is_nothrow_constructible< _Tp, _Args >`, 3111  
`std::is_nothrow_copy_assignable< _Tp >`, 3112  
`std::is_nothrow_copy_constructible< _Tp >`, 3113



- `std::is_nothrow_default_constructible< _Tp >`, 3113
- `std::is_nothrow_destructible< _Tp >`, 3114
- `std::is_nothrow_move_assignable< _Tp >`, 3114
- `std::is_nothrow_move_constructible< _Tp >`, 3115
- `std::is_nothrow_swappable< _Tp >`, 3115
- `std::is_nothrow_swappable_with< _Tp, _Up >`, 3116
- `std::is_null_pointer< _Tp >`, 3116
- `std::is_object< _Tp >`, 3117
- `std::is_placeholder< _Placeholder< _Num > >`, 3119
- `std::is_placeholder< _Tp >`, 3118
- `std::is_pod< _Tp >`, 3120
- `std::is_pointer< _Tp >`, 3121
- `std::is_polymorphic< _Tp >`, 3121
- `std::is_reference< _Tp >`, 3122
- `std::is_rvalue_reference< typename >`, 3122
- `std::is_same< typename, typename >`, 3123
- `std::is_scalar< _Tp >`, 3124
- `std::is_standard_layout< _Tp >`, 3125
- `std::is_swappable< _Tp >`, 3126
- `std::is_swappable_with< _Tp, _Up >`, 3126
- `std::is_trivial< _Tp >`, 3127
- `std::is_trivially_assignable< _Tp, _Up >`, 3128
- `std::is_trivially_constructible< _Tp, _Args >`, 3129
- `std::is_trivially_default_constructible< _Tp >`, 3129
- `std::is_trivially_destructible< _Tp >`, 3129
- `std::is_union< _Tp >`, 3130
- `std::is_void< _Tp >`, 3131
- `std::is_volatile< typename >`, 3132
- `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`, 3133
  - difference\_type, 3134
  - istream\_iterator, 3135
  - iterator\_category, 3134
  - pointer, 3134
  - reference, 3135
  - value\_type, 3135
- `std::istreambuf_iterator< _CharT, _Traits >`, 3136
  - char\_type, 3137
  - difference\_type, 3137
  - equal, 3140
  - int\_type, 3138
  - istream\_type, 3138
  - istreambuf\_iterator, 3139, 3140
  - iterator\_category, 3138
  - operator\*, 3140
  - operator++, 3141
  - pointer, 3138
  - reference, 3138
  - streambuf\_type, 3139
  - traits\_type, 3139
  - value\_type, 3139
- `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`, 3141
  - difference\_type, 3142
  - iterator\_category, 3142
  - pointer, 3142
  - reference, 3143
  - value\_type, 3143
- `std::iterator_traits< _Tp * >`, 3143
- `std::iterator_traits< const _Tp * >`, 3144
- `std::length_error`, 3145
  - what, 3145
- `std::less< _Tp >`, 3146
  - first\_argument\_type, 3146
  - result\_type, 3147
  - second\_argument\_type, 3147
- `std::less< void >`, 3147
- `std::less_equal< _Tp >`, 3148
  - first\_argument\_type, 3149
  - result\_type, 3149
  - second\_argument\_type, 3149
- `std::less_equal< void >`, 3149
- `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, 3150
  - discard, 3152
  - increment, 3156
  - linear\_congruential\_engine, 3151, 3152
  - max, 3152
  - min, 3153
  - modulus, 3156
  - multiplier, 3156
  - operator<<, 3154
  - operator>>, 3155
  - operator(), 3153
  - operator==, 3155
  - result\_type, 3151
  - seed, 3153, 3154
- `std::list< _Tp, _Alloc >`, 3157
  - \_M\_create\_node, 3164
  - ~list, 3164
  - assign, 3164, 3165
  - back, 3166
  - begin, 3166, 3167
  - cbegin, 3167
  - cend, 3167
  - clear, 3167
  - crbegin, 3168
  - crend, 3168
  - emplace, 3168
  - empty, 3169
  - end, 3169
  - erase, 3169, 3170
  - front, 3170, 3171
  - get\_allocator, 3171
  - insert, 3171–3173
  - list, 3161–3163
  - max\_size, 3174
  - merge, 3174, 3175
  - operator=, 3175, 3176

- pop\_back, [3177](#)
- pop\_front, [3177](#)
- push\_back, [3177](#)
- push\_front, [3178](#)
- rbegin, [3178](#)
- remove, [3178](#)
- remove\_if, [3179](#)
- rend, [3179](#)
- resize, [3180](#)
- reverse, [3180](#)
- size, [3181](#)
- sort, [3181](#)
- splice, [3181–3183](#)
- swap, [3184](#)
- unique, [3184](#)
- std::locale, [3185](#)
  - ~locale, [3190](#)
  - all, [3195](#)
  - category, [3187](#)
  - classic, [3191](#)
  - collate, [3195](#)
  - combine, [3191](#)
  - ctype, [3195](#)
  - global, [3191](#)
  - has\_facet, [3194](#)
  - locale, [3187–3190](#)
  - messages, [3196](#)
  - monetary, [3196](#)
  - name, [3192](#)
  - none, [3196](#)
  - numeric, [3196](#)
  - operator!=, [3192](#)
  - operator(), [3192](#)
  - operator=, [3193](#)
  - operator==, [3193](#)
  - time, [3197](#)
  - use\_facet, [3194](#)
- std::locale::facet, [3198](#)
  - ~facet, [3200](#)
  - facet, [3199](#)
- std::locale::id, [3200](#)
  - has\_facet, [3201](#)
  - id, [3200](#)
  - use\_facet, [3201](#)
- std::lock\_guard< \_Mutex >, [3202](#)
- std::logic\_error, [3203](#)
  - logic\_error, [3203](#)
  - what, [3204](#)
- std::logical\_and< \_Tp >, [3204](#)
  - first\_argument\_type, [3205](#)
  - result\_type, [3205](#)
  - second\_argument\_type, [3205](#)
- std::logical\_and< void >, [3206](#)
- std::logical\_not< \_Tp >, [3206](#)
  - argument\_type, [3207](#)
  - result\_type, [3207](#)
- std::logical\_not< void >, [3208](#)
- std::logical\_or< \_Tp >, [3208](#)
  - first\_argument\_type, [3209](#)
  - result\_type, [3209](#)
  - second\_argument\_type, [3209](#)
- std::logical\_or< void >, [3210](#)
- std::lognormal\_distribution< \_RealType >, [3210](#)
  - max, [3212](#)
  - min, [3212](#)
  - operator<<, [3213](#)
  - operator>>, [3214](#)
  - operator(), [3212](#)
  - operator==, [3214](#)
  - param, [3212, 3213](#)
  - reset, [3213](#)
  - result\_type, [3212](#)
- std::lognormal\_distribution< \_RealType >::param\_type, [3215](#)
- std::make\_signed< \_Tp >, [3215](#)
- std::make\_unsigned< \_Tp >, [3216](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3216](#)
  - ~map, [3224](#)
  - at, [3224](#)
  - begin, [3225](#)
  - cbegin, [3225](#)
  - cend, [3225](#)
  - clear, [3226](#)
  - count, [3226](#)
  - crbegin, [3227](#)
  - crend, [3227](#)
  - emplace, [3227](#)
  - emplace\_hint, [3229](#)
  - empty, [3229](#)
  - end, [3230](#)
  - equal\_range, [3230–3232](#)
  - erase, [3233, 3234](#)
  - find, [3235, 3236](#)
  - get\_allocator, [3237](#)
  - insert, [3237–3241](#)
  - key\_comp, [3242](#)
  - lower\_bound, [3242, 3243](#)
  - map, [3220–3223](#)
  - max\_size, [3244](#)
  - operator=, [3244, 3245](#)
  - operator[], [3245](#)
  - rbegin, [3246](#)
  - rend, [3246](#)
  - size, [3247](#)
  - swap, [3247](#)
  - upper\_bound, [3247–3249](#)
  - value\_comp, [3249](#)
- std::mask\_array< \_Tp >, [3250](#)

std::match\_results< \_Bi\_iter, \_Alloc >, 3252  
     ~match\_results, 3257  
     begin, 3257  
     cbegin, 3257  
     cend, 3258  
     empty, 3258  
     end, 3258  
     format, 3258, 3259  
     get\_allocator, 3260  
     length, 3260  
     match\_results, 3256, 3257  
     max\_size, 3260  
     operator=, 3261  
     operator[], 3261  
     position, 3262  
     prefix, 3262  
     ready, 3262  
     size, 3263  
     str, 3263  
     suffix, 3264  
     swap, 3264  
 std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, 3265  
     first\_argument\_type, 3265  
     result\_type, 3266  
     second\_argument\_type, 3266  
 std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, 3266  
     first\_argument\_type, 3267  
     result\_type, 3267  
     second\_argument\_type, 3267  
 std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 3268  
     argument\_type, 3269  
     result\_type, 3269  
 std::mem\_fun\_t< \_Ret, \_Tp >, 3269  
     argument\_type, 3270  
     result\_type, 3270  
 std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n,  
     \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l,  
     \_\_f >, 3270  
     discard, 3273  
     max, 3274  
     mersenne\_twister\_engine, 3273  
     min, 3274  
     operator<<, 3274  
     operator>>, 3275  
     operator==, 3275  
     result\_type, 3272  
 std::messages< \_CharT >, 3276  
     ~messages, 3279  
     char\_type, 3278  
     do\_get, 3279  
     id, 3280  
     messages, 3278, 3279  
     string\_type, 3278  
 std::messages\_base, 3280  
 std::messages\_byname< \_CharT >, 3281  
     do\_get, 3283  
     id, 3283  
 std::minus< \_Tp >, 3284  
     first\_argument\_type, 3284  
     result\_type, 3285  
     second\_argument\_type, 3285  
 std::minus< void >, 3285  
 std::modulus< \_Tp >, 3286  
     first\_argument\_type, 3287  
     result\_type, 3287  
     second\_argument\_type, 3287  
 std::modulus< void >, 3287  
 std::money\_base, 3288  
 std::money\_get< \_CharT, \_InIter >, 3289  
     ~money\_get, 3292  
     char\_type, 3291  
     do\_get, 3292  
     get, 3293, 3294  
     id, 3295  
     iter\_type, 3291  
     money\_get, 3291  
     string\_type, 3291  
 std::money\_put< \_CharT, \_OutIter >, 3295  
     ~money\_put, 3298  
     char\_type, 3297  
     do\_put, 3298, 3299  
     id, 3301  
     iter\_type, 3297  
     money\_put, 3297  
     put, 3299, 3300  
     string\_type, 3297  
 std::moneypunct< \_CharT, \_Intl >, 3302  
     ~moneypunct, 3305  
     char\_type, 3304  
     curr\_symbol, 3306  
     decimal\_point, 3306  
     do\_curr\_symbol, 3306  
     do\_decimal\_point, 3307  
     do\_frac\_digits, 3307  
     do\_grouping, 3308  
     do\_neg\_format, 3308  
     do\_negative\_sign, 3309  
     do\_pos\_format, 3309  
     do\_positive\_sign, 3310  
     do\_thousands\_sep, 3310  
     frac\_digits, 3311  
     grouping, 3311  
     id, 3314  
     intl, 3315  
     moneypunct, 3304, 3305  
     neg\_format, 3312  
     negative\_sign, 3312  
     pos\_format, 3313

- positive\_sign, 3313
- string\_type, 3304
- thousands\_sep, 3314
- std::moneypunct\_byname< \_CharT, \_Intl >, 3315
  - curr\_symbol, 3317
  - decimal\_point, 3318
  - do\_curr\_symbol, 3318
  - do\_decimal\_point, 3318
  - do\_frac\_digits, 3319
  - do\_grouping, 3319
  - do\_neg\_format, 3320
  - do\_negative\_sign, 3320
  - do\_pos\_format, 3321
  - do\_positive\_sign, 3321
  - do\_thousands\_sep, 3322
  - frac\_digits, 3322
  - grouping, 3323
  - id, 3326
  - neg\_format, 3323
  - negative\_sign, 3324
  - pos\_format, 3324
  - positive\_sign, 3325
  - thousands\_sep, 3325
- std::move\_iterator< \_Iterator >, 3326
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3328
  - ~multimap, 3335
  - begin, 3335
  - cbegin, 3336
  - cend, 3336
  - clear, 3336
  - count, 3336, 3337
  - crbegin, 3337
  - crend, 3337
  - emplace, 3338
  - emplace\_hint, 3338
  - empty, 3339
  - end, 3339
  - equal\_range, 3340, 3341
  - erase, 3342–3344
  - find, 3344–3346
  - get\_allocator, 3346
  - insert, 3346–3351
  - key\_comp, 3351
  - lower\_bound, 3351–3353
  - max\_size, 3353
  - multimap, 3331–3334
  - operator=, 3353, 3354
  - rbegin, 3354, 3355
  - rend, 3355
  - size, 3355
  - swap, 3356
  - upper\_bound, 3356–3358
  - value\_comp, 3358
- std::multiplies< \_Tp >, 3359
- first\_argument\_type, 3359
- result\_type, 3360
- second\_argument\_type, 3360
- std::multiplies< void >, 3360
- std::multiset< \_Key, \_Compare, \_Alloc >, 3361
  - ~multiset, 3368
  - begin, 3368
  - cbegin, 3368
  - cend, 3368
  - clear, 3369
  - count, 3369
  - crbegin, 3370
  - crend, 3370
  - emplace, 3370
  - emplace\_hint, 3371
  - empty, 3371
  - end, 3372
  - equal\_range, 3372–3374
  - erase, 3374, 3375
  - find, 3376, 3377
  - get\_allocator, 3378
  - insert, 3378–3380
  - key\_comp, 3380
  - lower\_bound, 3380–3382
  - max\_size, 3382
  - multiset, 3364–3367
  - operator=, 3382, 3383
  - rbegin, 3383
  - rend, 3384
  - size, 3384
  - swap, 3384
  - upper\_bound, 3385, 3386
  - value\_comp, 3386
- std::mutex, 3387
- std::negate< \_Tp >, 3388
  - argument\_type, 3388
  - result\_type, 3388
- std::negate< void >, 3389
- std::negative\_binomial\_distribution< \_IntType >, 3389
  - k, 3391
  - max, 3391
  - min, 3391
  - operator<<, 3393
  - operator>>, 3393
  - operator(), 3391
  - operator==, 3393
  - p, 3392
  - param, 3392
  - reset, 3393
  - result\_type, 3391
- std::negative\_binomial\_distribution< \_IntType >::param\_type, 3394
- std::nested\_exception, 3395
- std::normal\_distribution< \_RealType >, 3395

- max, [3397](#)
- mean, [3397](#)
- min, [3398](#)
- normal\_distribution, [3397](#)
- operator<=, [3400](#)
- operator>, [3400](#)
- operator(), [3398](#)
- operator==, [3400](#)
- param, [3398](#), [3399](#)
- reset, [3399](#)
- result\_type, [3397](#)
- stddev, [3399](#)
- std::normal\_distribution< \_RealType >::param\_type, [3401](#)
- std::not\_equal\_to< \_Tp >, [3402](#)
  - first\_argument\_type, [3402](#)
  - result\_type, [3403](#)
  - second\_argument\_type, [3403](#)
- std::not\_equal\_to< void >, [3403](#)
- std::num\_get< \_CharT, \_InIter >, [3404](#)
  - ~num\_get, [3407](#)
  - char\_type, [3406](#)
  - do\_get, [3408–3415](#)
  - get, [3415–3424](#)
  - id, [3425](#)
  - iter\_type, [3407](#)
  - num\_get, [3407](#)
- std::num\_put< \_CharT, \_OutIter >, [3425](#)
  - ~num\_put, [3429](#)
  - char\_type, [3427](#)
  - do\_put, [3429–3433](#)
  - id, [3442](#)
  - iter\_type, [3427](#)
  - num\_put, [3428](#)
  - put, [3434–3441](#)
- std::numeric\_limits< \_Tp >, [3442](#)
  - denorm\_min, [3444](#)
  - digits, [3446](#)
  - digits10, [3446](#)
  - epsilon, [3444](#)
  - has\_denorm, [3446](#)
  - has\_denorm\_loss, [3446](#)
  - has\_infinity, [3446](#)
  - has\_quiet\_NaN, [3447](#)
  - has\_signaling\_NaN, [3447](#)
  - infinity, [3444](#)
  - is\_bounded, [3447](#)
  - is\_exact, [3447](#)
  - is\_iec559, [3447](#)
  - is\_integer, [3448](#)
  - is\_modulo, [3448](#)
  - is\_signed, [3448](#)
  - is\_specialized, [3448](#)
  - lowest, [3444](#)
  - max, [3444](#)
  - max\_digits10, [3448](#)
  - max\_exponent, [3449](#)
  - max\_exponent10, [3449](#)
  - min, [3445](#)
  - min\_exponent, [3449](#)
  - min\_exponent10, [3449](#)
  - quiet\_NaN, [3445](#)
  - radix, [3449](#)
  - round\_error, [3445](#)
  - round\_style, [3450](#)
  - signaling\_NaN, [3445](#)
  - tinyness\_before, [3450](#)
  - traps, [3450](#)
- std::numeric\_limits< bool >, [3451](#)
- std::numeric\_limits< char >, [3452](#)
- std::numeric\_limits< char16\_t >, [3453](#)
- std::numeric\_limits< char32\_t >, [3454](#)
- std::numeric\_limits< double >, [3455](#)
- std::numeric\_limits< float >, [3456](#)
- std::numeric\_limits< int >, [3457](#)
- std::numeric\_limits< long >, [3458](#)
- std::numeric\_limits< long double >, [3459](#)
- std::numeric\_limits< long long >, [3460](#)
- std::numeric\_limits< short >, [3461](#)
- std::numeric\_limits< signed char >, [3462](#)
- std::numeric\_limits< unsigned char >, [3463](#)
- std::numeric\_limits< unsigned int >, [3464](#)
- std::numeric\_limits< unsigned long >, [3465](#)
- std::numeric\_limits< unsigned long long >, [3466](#)
- std::numeric\_limits< unsigned short >, [3467](#)
- std::numeric\_limits< wchar\_t >, [3468](#)
- std::num\_punct< \_CharT >, [3469](#)
  - ~num\_punct, [3472](#)
  - char\_type, [3470](#)
  - decimal\_point, [3472](#)
  - do\_decimal\_point, [3473](#)
  - do\_falsename, [3473](#)
  - do\_grouping, [3473](#)
  - do\_thousands\_sep, [3474](#)
  - do\_truename, [3474](#)
  - falsename, [3475](#)
  - grouping, [3475](#)
  - id, [3477](#)
  - num\_punct, [3471](#), [3472](#)
  - string\_type, [3471](#)
  - thousands\_sep, [3476](#)
  - truename, [3476](#)
- std::num\_punct\_byname< \_CharT >, [3477](#)
  - decimal\_point, [3479](#)
  - do\_decimal\_point, [3479](#)
  - do\_falsename, [3479](#)
  - do\_grouping, [3480](#)
  - do\_thousands\_sep, [3480](#)

- do\_truename, 3481
- falsename, 3481
- grouping, 3481
- id, 3483
- thousands\_sep, 3482
- truename, 3482
- std::once\_flag, 3483
  - call\_once, 3484
  - once\_flag, 3484
  - operator=, 3484
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3485
  - char\_type, 3486
  - difference\_type, 3486
  - iterator\_category, 3487
  - operator=, 3489
  - ostream\_iterator, 3488, 3489
  - ostream\_type, 3487
  - pointer, 3487
  - reference, 3487
  - traits\_type, 3487
  - value\_type, 3488
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 3490
  - char\_type, 3491
  - difference\_type, 3491
  - failed, 3493
  - iterator\_category, 3491
  - operator\*, 3493
  - operator++, 3494
  - operator=, 3494
  - ostream\_type, 3491
  - ostreambuf\_iterator, 3493
  - pointer, 3492
  - reference, 3492
  - streambuf\_type, 3492
  - traits\_type, 3492
  - value\_type, 3492
- std::out\_of\_range, 3495
  - what, 3495
- std::output\_iterator\_tag, 3496
- std::overflow\_error, 3496
  - what, 3497
- std::owner\_less< \_Tp >, 3497
- std::owner\_less< shared\_ptr< \_Tp > >, 3497
  - first\_argument\_type, 3498
  - result\_type, 3498
  - second\_argument\_type, 3498
- std::owner\_less< void >, 3498
  - first\_argument\_type, 3499
  - result\_type, 3499
  - second\_argument\_type, 3499
- std::owner\_less< weak\_ptr< \_Tp > >, 3500
  - first\_argument\_type, 3500
  - result\_type, 3501
  - second\_argument\_type, 3501
- std::packaged\_task< \_Res(\_ArgTypes...)>, 3501
- std::pair< \_T1, \_T2 >, 3502
  - \_PCCFP, 3504
  - \_PCCP, 3504
  - first, 3505
  - pair, 3504
  - second, 3505
  - second\_type, 3504
- std::piecewise\_constant\_distribution< \_RealType >, 3506
  - densities, 3507
  - intervals, 3507
  - max, 3508
  - min, 3508
  - operator<=, 3509
  - operator>=, 3510
  - operator(), 3508
  - operator==, 3510
  - param, 3508, 3509
  - reset, 3509
  - result\_type, 3507
- std::piecewise\_constant\_distribution< \_RealType >::param\_type, 3511
- std::piecewise\_construct\_t, 3511
- std::piecewise\_linear\_distribution< \_RealType >, 3512
  - densities, 3513
  - intervals, 3513
  - max, 3514
  - min, 3514
  - operator<=, 3515
  - operator>=, 3516
  - operator(), 3514
  - operator==, 3516
  - param, 3514, 3515
  - reset, 3515
  - result\_type, 3513
- std::piecewise\_linear\_distribution< \_RealType >::param\_type, 3517
- std::placeholders, 838
- std::plus< \_Tp >, 3518
  - first\_argument\_type, 3518
  - result\_type, 3519
  - second\_argument\_type, 3519
- std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, 3519
  - first\_argument\_type, 3520
  - result\_type, 3520
  - second\_argument\_type, 3520
- std::pointer\_to\_unary\_function< \_Arg, \_Result >, 3521
  - argument\_type, 3522
  - result\_type, 3522
- std::pointer\_traits< \_Ptr >, 3522
  - difference\_type, 3523
  - element\_type, 3523
  - pointer, 3523

- rebind, 3523
- std::pointer\_traits< \_Tp \* >, 3524
  - difference\_type, 3524
  - element\_type, 3525
  - pointer, 3525
  - pointer\_to, 3525
- std::poisson\_distribution< \_IntType >, 3526
  - max, 3527
  - mean, 3527
  - min, 3528
  - operator<=, 3529
  - operator>, 3530
  - operator(), 3528
  - operator==, 3530
  - param, 3528, 3529
  - reset, 3529
  - result\_type, 3527
- std::poisson\_distribution< \_IntType >::param\_type, 3531
- std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 3531
  - empty, 3534
  - pop, 3534
  - priority\_queue, 3533
  - push, 3535
  - size, 3535
  - top, 3535
- std::promise< \_Res >, 3536
- std::promise< \_Res & >, 3537
- std::promise< void >, 3537
- std::queue< \_Tp, \_Sequence >, 3538
  - back, 3540
  - c, 3542
  - empty, 3540
  - front, 3540
  - pop, 3541
  - push, 3541
  - queue, 3539
  - size, 3541
- std::random\_access\_iterator\_tag, 3543
- std::random\_device, 3543
  - result\_type, 3544
- std::range\_error, 3545
  - what, 3545
- std::rank< typename >, 3546
- std::ratio< \_Num, \_Den >, 3547
- std::ratio\_equal< \_R1, \_R2 >, 3547
- std::ratio\_not\_equal< \_R1, \_R2 >, 3548
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 3549
  - difference\_type, 3550
  - iterator\_category, 3550
  - pointer, 3551
  - reference, 3551
  - value\_type, 3551
- std::recursive\_mutex, 3551
- std::recursive\_timed\_mutex, 3552
- std::reference\_wrapper< \_Tp >, 3553
- std::regex\_constants, 839
  - \_\_match\_flag, 840
  - \_\_polynomial, 848
  - \_\_syntax\_option, 841
  - awk, 849
  - basic, 849
  - collate, 849
  - ECMAScript, 849
  - egrep, 850
  - error\_backref, 842
  - error\_badbrace, 842
  - error\_badrepeat, 842
  - error\_brace, 842
  - error\_brack, 842
  - error\_collate, 843
  - error\_complexity, 843
  - error\_ctype, 843
  - error\_escape, 843
  - error\_paren, 843
  - error\_range, 843
  - error\_space, 843
  - error\_stack, 844
  - error\_type, 841
  - extended, 850
  - format\_default, 850
  - format\_first\_only, 851
  - format\_no\_copy, 851
  - format\_sed, 851
  - grep, 851
  - icase, 852
  - match\_any, 852
  - match\_continuous, 852
  - match\_default, 852
  - match\_flag\_type, 841
  - match\_not\_bol, 852
  - match\_not\_bow, 853
  - match\_not\_eol, 853
  - match\_not\_eow, 853
  - match\_not\_null, 853
  - match\_prev\_avail, 853
  - nosubs, 854
  - operator~, 848
  - operator^, 845
  - operator^=, 846
  - operator&, 844
  - operator&=, 844, 845
  - operator|, 846, 847
  - operator|=, 847
  - optimize, 854
  - syntax\_option\_type, 841
- std::regex\_error, 3554
  - code, 3555
  - regex\_error, 3554

- what, [3555](#)
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3555](#)
  - operator!=, [3557](#)
  - operator\*, [3557](#)
  - operator++, [3557](#), [3558](#)
  - operator->, [3558](#)
  - operator=, [3558](#)
  - operator==, [3558](#)
  - regex\_iterator, [3556](#), [3557](#)
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3559](#)
  - operator!=, [3562](#)
  - operator\*, [3563](#)
  - operator++, [3563](#)
  - operator->, [3563](#)
  - operator=, [3564](#)
  - operator==, [3564](#)
  - regex\_token\_iterator, [3560–3562](#)
- std::regex\_traits< \_Ch\_type >, [3564](#)
  - getloc, [3566](#)
  - imbue, [3566](#)
  - isctype, [3566](#)
  - length, [3567](#)
  - lookup\_classname, [3567](#)
  - lookup\_collatename, [3568](#)
  - regex\_traits, [3565](#)
  - transform, [3569](#)
  - transform\_primary, [3570](#)
  - translate, [3570](#)
  - translate\_nocase, [3571](#)
  - value, [3571](#)
- std::rel\_ops, [854](#)
  - operator!=, [855](#)
  - operator<=, [856](#)
  - operator>, [856](#)
  - operator>=, [857](#)
- std::remove\_all\_extents< \_Tp >, [3572](#)
- std::remove\_const< \_Tp >, [3572](#)
- std::remove\_cv< \_Tp >, [3573](#)
- std::remove\_extent< \_Tp >, [3573](#)
- std::remove\_pointer< \_Tp >, [3573](#)
- std::remove\_reference< \_Tp >, [3574](#)
- std::remove\_volatile< \_Tp >, [3574](#)
- std::result\_of< \_Signature >, [3575](#)
- std::reverse\_iterator< \_Iterator >, [3575](#)
  - base, [3578](#)
  - iterator\_category, [3577](#)
  - operator\*, [3578](#)
  - operator+, [3579](#)
  - operator++, [3579](#)
  - operator+=, [3580](#)
  - operator-, [3580](#)
  - operator->, [3581](#)
  - operator--, [3580](#), [3581](#)
  - operator=, [3581](#)
  - operator[], [3582](#)
  - reverse\_iterator, [3577](#), [3578](#)
  - value\_type, [3577](#)
- std::runtime\_error, [3582](#)
  - runtime\_error, [3583](#)
  - what, [3583](#)
- std::scoped\_allocator\_adaptor< \_OuterAlloc, \_InnerAllocs >, [3583](#)
- std::seed\_seq, [3585](#)
  - result\_type, [3586](#)
  - seed\_seq, [3586](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3587](#)
  - ~set, [3597](#)
  - allocator\_type, [3590](#)
  - begin, [3598](#)
  - cbegin, [3598](#)
  - cend, [3598](#)
  - clear, [3598](#)
  - const\_iterator, [3590](#)
  - const\_pointer, [3590](#)
  - const\_reference, [3590](#)
  - const\_reverse\_iterator, [3591](#)
  - count, [3598](#), [3599](#)
  - crbegin, [3599](#)
  - crend, [3599](#)
  - difference\_type, [3591](#)
  - emplace, [3600](#)
  - emplace\_hint, [3600](#)
  - empty, [3601](#)
  - end, [3601](#)
  - equal\_range, [3601–3603](#)
  - erase, [3604](#), [3605](#)
  - find, [3605–3607](#)
  - get\_allocator, [3607](#)
  - insert, [3607–3609](#)
  - iterator, [3591](#)
  - key\_comp, [3610](#)
  - key\_compare, [3591](#)
  - key\_type, [3592](#)
  - lower\_bound, [3610](#), [3611](#)
  - max\_size, [3612](#)
  - operator=, [3612](#), [3613](#)
  - pointer, [3592](#)
  - rbegin, [3613](#)
  - reference, [3592](#)
  - rend, [3613](#)
  - reverse\_iterator, [3592](#)
  - set, [3593–3597](#)
  - size, [3614](#)
  - size\_type, [3593](#)
  - swap, [3614](#)
  - upper\_bound, [3614–3616](#)



- value\_comp, 3616
- value\_compare, 3593
- value\_type, 3593
- std::shared\_future< \_Res >, 3617
  - \_M\_get\_result, 3619
  - \_Ptr, 3618
  - get, 3619
  - shared\_future, 3619
- std::shared\_future< \_Res & >, 3620
  - \_M\_get\_result, 3623
  - \_Ptr, 3622
  - get, 3623
  - shared\_future, 3622
- std::shared\_future< void >, 3623
  - \_M\_get\_result, 3626
  - \_Ptr, 3625
  - shared\_future, 3625
- std::shared\_lock< \_Mutex >, 3626
- std::shared\_ptr< \_Tp >, 3627
  - allocate\_shared, 3635
  - shared\_ptr, 3628–3635
- std::shared\_timed\_mutex, 3636
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3637
  - base, 3640
  - discard, 3640
  - max, 3641
  - min, 3641
  - operator<<, 3642
  - operator>>, 3643
  - operator(), 3641
  - operator==, 3643
  - result\_type, 3638
  - seed, 3641, 3642
  - shuffle\_order\_engine, 3638–3640
- std::slice, 3644
- std::slice\_array< \_Tp >, 3644
- std::stack< \_Tp, \_Sequence >, 3646
  - empty, 3648
  - pop, 3648
  - push, 3648
  - size, 3648
  - stack, 3647
  - top, 3649
- std::student\_t\_distribution< \_RealType >, 3649
  - max, 3651
  - min, 3651
  - operator<<, 3653
  - operator>>, 3653
  - operator(), 3651
  - operator==, 3653
  - param, 3652
  - reset, 3652
  - result\_type, 3651
- std::student\_t\_distribution< \_RealType >::param\_type, 3654
- std::sub\_match< \_Biter >, 3655
  - \_PCCFP, 3656
  - \_PCCP, 3656
  - compare, 3656, 3657
  - first, 3659
  - length, 3658
  - operator string\_type, 3658
  - second, 3659
  - second\_type, 3656
  - str, 3658
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3659
  - discard, 3662
  - max, 3662
  - min, 3662
  - operator<<, 3663
  - operator>>, 3664
  - operator(), 3662
  - operator==, 3664
  - result\_type, 3661
  - seed, 3662, 3663
  - subtract\_with\_carry\_engine, 3661
- std::system\_error, 3665
  - what, 3666
- std::this\_thread, 857
  - get\_id, 858
  - sleep\_for, 858
  - sleep\_until, 858
  - yield, 858
- std::thread, 3666
  - native\_handle, 3667
- std::thread::id, 3667
- std::time\_base, 3668
- std::time\_get< \_CharT, \_InIter >, 3669
  - ~time\_get, 3671
  - char\_type, 3671
  - date\_order, 3672
  - do\_date\_order, 3672
  - do\_get, 3672
  - do\_get\_date, 3673
  - do\_get\_monthname, 3674
  - do\_get\_time, 3675
  - do\_get\_weekday, 3676
  - do\_get\_year, 3676
  - get, 3677, 3678
  - get\_date, 3679
  - get\_monthname, 3679
  - get\_time, 3680
  - get\_weekday, 3681
  - get\_year, 3682
  - id, 3682
  - iter\_type, 3671

- time\_get, 3671
- std::time\_get\_byname< \_CharT, \_InIter >, 3683
  - date\_order, 3685
  - do\_date\_order, 3685
  - do\_get, 3685
  - do\_get\_date, 3686
  - do\_get\_monthname, 3687
  - do\_get\_time, 3688
  - do\_get\_weekday, 3689
  - do\_get\_year, 3689
  - get, 3690, 3691
  - get\_date, 3692
  - get\_monthname, 3692
  - get\_time, 3693
  - get\_weekday, 3694
  - get\_year, 3695
  - id, 3695
- std::time\_put< \_CharT, \_OutIter >, 3696
  - ~time\_put, 3698
  - char\_type, 3697
  - do\_put, 3699
  - id, 3701
  - iter\_type, 3698
  - put, 3699, 3700
  - time\_put, 3698
- std::time\_put\_byname< \_CharT, \_OutIter >, 3702
  - do\_put, 3703
  - id, 3705
  - put, 3704
- std::timed\_mutex, 3706
- std::tr1, 859
  - conf\_hyperg, 861
  - hyperg, 862
- std::tr1::\_\_detail, 862
- std::tr2, 862
- std::tr2::\_\_detail, 864
- std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc >, 3706
  - \_M\_w, 3708
- std::tr2::\_\_reflection\_typelist< \_Elements >, 3708
- std::tr2::\_\_reflection\_typelist< \_First, \_Rest... >, 3709
- std::tr2::\_\_reflection\_typelist<>, 3709
- std::tr2::bases< \_Tp >, 3709
- std::tr2::bool\_set, 3710
  - bool\_set, 3711
  - equals, 3711
  - is\_emptyset, 3711
  - is\_indeterminate, 3712
  - is\_singleton, 3712
  - operator bool, 3712
- std::tr2::direct\_bases< \_Tp >, 3712
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3713
  - all, 3720
  - any, 3720
  - append, 3720, 3721
  - clear, 3721
  - count, 3721
  - dynamic\_bitset, 3717–3719
  - empty, 3721
  - find\_first, 3722
  - find\_next, 3722
  - flip, 3723
  - get\_allocator, 3723
  - max\_size, 3723
  - none, 3724
  - num\_blocks, 3724
  - operator<<, 3725
  - operator<=, 3725
  - operator>>, 3726
  - operator>=, 3726
  - operator~, 3728
  - operator^=, 3728
  - operator-=, 3725
  - operator=, 3726
  - operator&=, 3724, 3725
  - operator[], 3727
  - operator|=, 3728
  - push\_back, 3729
  - reset, 3729
  - resize, 3730
  - set, 3730
  - size, 3731
  - swap, 3731
  - test, 3731
  - to\_string, 3732
  - to\_ullong, 3732
  - to\_ulong, 3732
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::reference, 3733
- std::try\_to\_lock\_t, 3734
- std::tuple< \_Elements >, 3734
- std::tuple< \_T1, \_T2 >, 3737
- std::tuple\_element< 0, std::pair< \_Tp1, \_Tp2 > >, 3740
- std::tuple\_element< 0, tuple< \_Head, \_Tail... > >, 3740
- std::tuple\_element< 1, std::pair< \_Tp1, \_Tp2 > >, 3740
- std::tuple\_element< \_\_i, tuple< \_Head, \_Tail... > >, 3741
- std::tuple\_element< \_\_i, tuple<> >, 3742
- std::tuple\_element< \_Int, ::array< \_Tp, \_Nm > >, 3742
- std::tuple\_element< \_Int, \_Tp >, 3739
- std::tuple\_element< \_Int, std::\_\_debug::array< \_Tp, \_Nm > >, 3742
- std::tuple\_size< \_Tp >, 3743
- std::tuple\_size< std::\_\_debug::array< \_Tp, \_Nm > >, 3743
- std::tuple\_size< std::pair< \_Tp1, \_Tp2 > >, 3744
- std::tuple\_size< tuple< \_Elements... > >, 3745
- std::tuple\_size<::array< \_Tp, \_Nm > >, 3746
- std::type\_index, 3747
- std::type\_info, 3748

~type\_info, 3749  
 name, 3749  
 std::unary\_function<\_Arg, \_Result >, 3749  
   argument\_type, 3750  
   result\_type, 3750  
 std::unary\_negate<\_Predicate >, 3751  
   argument\_type, 3751  
   result\_type, 3752  
 std::underflow\_error, 3752  
   what, 3753  
 std::underlying\_type<\_Tp >, 3753  
 std::uniform\_int\_distribution<\_IntType >, 3753  
   max, 3755  
   min, 3755  
   operator(), 3755  
   operator==, 3757  
   param, 3756  
   reset, 3756  
   result\_type, 3754  
   uniform\_int\_distribution, 3755  
 std::uniform\_int\_distribution<\_IntType >::param\_type, 3757  
 std::uniform\_real\_distribution<\_RealType >, 3758  
   max, 3759  
   min, 3759  
   operator(), 3760  
   operator==, 3761  
   param, 3760  
   reset, 3761  
   result\_type, 3759  
   uniform\_real\_distribution, 3759  
 std::uniform\_real\_distribution<\_RealType >::param\_type, 3761  
 std::unique\_lock<\_Mutex >, 3762  
 std::unique\_ptr<\_Tp, \_Dp >, 3763  
   ~unique\_ptr, 3766  
   get, 3767  
   get\_deleter, 3767  
   operator bool, 3767  
   operator\*, 3768  
   operator->, 3768  
   operator=, 3768, 3769  
   release, 3769  
   reset, 3769  
   swap, 3770  
   unique\_ptr, 3764–3766  
 std::unique\_ptr<\_Tp[], \_Dp >, 3770  
   ~unique\_ptr, 3773  
   get, 3774  
   get\_deleter, 3774  
   operator bool, 3774  
   operator=, 3775, 3776  
   operator[], 3776  
   release, 3776  
   reset, 3776  
   swap, 3777  
   unique\_ptr, 3771–3773  
 std::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3777  
   allocator\_type, 3781  
   at, 3788  
   begin, 3789, 3790  
   bucket\_count, 3790  
   cbegin, 3790  
   cend, 3791  
   clear, 3792  
   const\_iterator, 3781  
   const\_local\_iterator, 3781  
   const\_pointer, 3782  
   const\_reference, 3782  
   count, 3792  
   difference\_type, 3782  
   emplace, 3792  
   emplace\_hint, 3793  
   empty, 3794  
   end, 3794, 3795  
   equal\_range, 3795, 3796  
   erase, 3796–3798  
   find, 3798, 3800  
   get\_allocator, 3800  
   hash\_function, 3800  
   hasher, 3782  
   insert, 3801–3805  
   iterator, 3783  
   key\_eq, 3805  
   key\_equal, 3783  
   key\_type, 3783  
   load\_factor, 3806  
   local\_iterator, 3783  
   mapped\_type, 3784  
   max\_bucket\_count, 3806  
   max\_load\_factor, 3806  
   max\_size, 3807  
   operator=, 3807  
   operator[], 3808  
   pointer, 3784  
   reference, 3784  
   rehash, 3809  
   reserve, 3809  
   size, 3810  
   size\_type, 3784  
   swap, 3810  
   unordered\_map, 3785–3787  
   value\_type, 3785  
 std::unordered\_multimap<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3811  
   allocator\_type, 3814  
   begin, 3821, 3822

- bucket\_count, 3822
- cbegin, 3822
- cend, 3823
- clear, 3824
- const\_iterator, 3814
- const\_local\_iterator, 3814
- const\_pointer, 3815
- const\_reference, 3815
- count, 3824
- difference\_type, 3815
- emplace, 3824
- emplace\_hint, 3825
- empty, 3825
- end, 3826, 3827
- equal\_range, 3827, 3828
- erase, 3828–3830
- find, 3830, 3831
- get\_allocator, 3831
- hash\_function, 3831
- hasher, 3815
- insert, 3832–3836
- iterator, 3816
- key\_eq, 3836
- key\_equal, 3816
- key\_type, 3816
- load\_factor, 3836
- local\_iterator, 3816
- mapped\_type, 3817
- max\_bucket\_count, 3836
- max\_load\_factor, 3837
- max\_size, 3837
- operator=, 3838
- pointer, 3817
- reference, 3817
- rehash, 3839
- reserve, 3839
- size, 3839
- size\_type, 3817
- swap, 3840
- unordered\_multimap, 3818–3820
- value\_type, 3818
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc  
>, 3840
  - allocator\_type, 3844
  - begin, 3850, 3851
  - bucket\_count, 3852
  - cbegin, 3852
  - cend, 3853
  - clear, 3853
  - const\_iterator, 3844
  - const\_local\_iterator, 3844
  - const\_pointer, 3844
  - const\_reference, 3845
  - count, 3854
  - difference\_type, 3845
  - emplace, 3854
  - emplace\_hint, 3855
  - empty, 3855
  - end, 3855, 3856
  - equal\_range, 3857
  - erase, 3858, 3860
  - find, 3862
  - get\_allocator, 3863
  - hash\_function, 3863
  - hasher, 3845
  - insert, 3863–3866
  - iterator, 3845
  - key\_eq, 3866
  - key\_equal, 3846
  - key\_type, 3846
  - load\_factor, 3867
  - local\_iterator, 3846
  - max\_bucket\_count, 3867
  - max\_load\_factor, 3867
  - max\_size, 3868
  - operator=, 3868
  - pointer, 3846
  - reference, 3847
  - rehash, 3869
  - reserve, 3869
  - size, 3870
  - size\_type, 3847
  - swap, 3870
  - unordered\_multiset, 3847–3850
  - value\_type, 3847
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3870
  - allocator\_type, 3874
  - begin, 3880, 3881
  - bucket\_count, 3882
  - cbegin, 3882
  - cend, 3883
  - clear, 3883
  - const\_iterator, 3874
  - const\_local\_iterator, 3874
  - const\_pointer, 3874
  - const\_reference, 3875
  - count, 3884
  - difference\_type, 3875
  - emplace, 3884
  - emplace\_hint, 3885
  - empty, 3885
  - end, 3886, 3887
  - equal\_range, 3887, 3888
  - erase, 3888–3890
  - find, 3890, 3891
  - get\_allocator, 3891
  - hash\_function, 3891

- hasher, [3875](#)
- insert, [3892–3895](#)
- iterator, [3875](#)
- key\_eq, [3895](#)
- key\_equal, [3876](#)
- key\_type, [3876](#)
- load\_factor, [3895](#)
- local\_iterator, [3876](#)
- max\_bucket\_count, [3895](#)
- max\_load\_factor, [3896](#)
- max\_size, [3896](#)
- operator=, [3897](#)
- pointer, [3876](#)
- reference, [3877](#)
- rehash, [3898](#)
- reserve, [3898](#)
- size, [3898](#)
- size\_type, [3877](#)
- swap, [3899](#)
- unordered\_set, [3877–3880](#)
- value\_type, [3877](#)
- std::uses\_allocator< \_Tp, \_Alloc >, [3899](#)
- std::uses\_allocator< tuple< \_Types... >, \_Alloc >, [3900](#)
- std::valarray< \_Tp >, [3901](#)
  - valarray, [3903](#)
- std::vector< \_Tp, \_Alloc >, [3903](#)
  - \_M\_allocate\_and\_copy, [3911](#)
  - \_M\_range\_check, [3911](#)
  - ~vector, [3911](#)
  - assign, [3911](#), [3912](#)
  - at, [3913](#)
  - back, [3915](#)
  - begin, [3915](#), [3916](#)
  - capacity, [3916](#)
  - cbegin, [3916](#)
  - cend, [3916](#)
  - clear, [3917](#)
  - crbegin, [3917](#)
  - crend, [3917](#)
  - data, [3917](#)
  - emplace, [3918](#)
  - empty, [3918](#)
  - end, [3918](#), [3919](#)
  - erase, [3919](#), [3920](#)
  - front, [3920](#)
  - get\_allocator, [3921](#)
  - insert, [3921–3923](#)
  - max\_size, [3924](#)
  - operator=, [3924](#), [3925](#)
  - operator[], [3925](#), [3926](#)
  - pop\_back, [3926](#)
  - push\_back, [3926](#)
  - rbegin, [3927](#)
  - rend, [3927](#)
  - reserve, [3928](#)
  - resize, [3928](#), [3929](#)
  - shrink\_to\_fit, [3929](#)
  - size, [3929](#)
  - swap, [3930](#)
  - vector, [3907–3910](#)
- std::vector< bool, \_Alloc >, [3930](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3934](#)
  - \_M\_buf\_locale, [3954](#)
  - \_M\_in\_beg, [3955](#)
  - \_M\_in\_cur, [3955](#)
  - \_M\_in\_end, [3955](#)
  - \_M\_out\_beg, [3955](#)
  - \_M\_out\_cur, [3955](#)
  - \_M\_out\_end, [3956](#)
  - \_\_streambuf\_type, [3937](#)
- char\_type, [3937](#)
- eback, [3939](#)
- egptr, [3939](#)
- epptr, [3939](#)
- gbump, [3940](#)
- getloc, [3940](#)
- gptr, [3940](#)
- imbue, [3941](#)
- in\_avail, [3941](#)
- int\_type, [3937](#)
- off\_type, [3937](#)
- overflow, [3942](#)
- pbackfail, [3942](#)
- pbase, [3943](#)
- pbump, [3943](#)
- pos\_type, [3938](#)
- pptr, [3944](#)
- pubimbue, [3944](#)
- pubseekoff, [3944](#)
- pubseekpos, [3945](#)
- pubsetbuf, [3945](#)
- pubsync, [3945](#)
- sbumpc, [3946](#)
- seekoff, [3946](#)
- seekpos, [3946](#)
- setbuf, [3947](#)
- setg, [3947](#)
- setp, [3948](#)
- sgetc, [3948](#)
- sgetn, [3948](#)
- showmanyc, [3949](#)
- snextc, [3949](#)
- sputbackc, [3950](#)
- sputc, [3950](#)
- sputn, [3951](#)
- state, [3951](#)
- sungetc, [3951](#)
- sync, [3952](#)

- traits\_type, 3938
- uflow, 3952
- underflow, 3953
- wbuffer\_convert, 3938
- xsggetn, 3953
- xspn, 3954
- std::weak\_ptr< \_Tp >, 3956
- std::weibull\_distribution< \_RealType >, 3957
  - a, 3958
  - b, 3958
  - max, 3959
  - min, 3959
  - operator(), 3959
  - operator==, 3960
  - param, 3959, 3960
  - reset, 3960
  - result\_type, 3958
- std::weibull\_distribution< \_RealType >::param\_type, 3961
- std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3961
  - converted, 3964
  - from\_bytes, 3964, 3965
  - state, 3965
  - to\_bytes, 3965, 3966
  - wstring\_convert, 3962, 3963
- std\_abs.h, 4334
- std\_function.h, 4334
- std\_mutex.h, 4335
- stdc++.h, 4336
- stddev
  - std::normal\_distribution< \_RealType >, 3399
- stdexcept, 4336
- stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 1045, 1046
- stdio\_filebuf.h, 4337
- stdio\_sync\_filebuf.h, 4337
- stdlib.h, 4337
- stdtr1c++.h, 4337
- stl\_algo.h, 4338
  - \_\_rotate, 4347, 4348
- stl\_algobase.h, 4348
- stl\_bvector.h, 4351
- stl\_construct.h, 4352
- stl\_deque.h, 4353
  - \_GLIBCXX\_DEQUE\_BUF\_SIZE, 4355
- stl\_function.h, 4355
- stl\_heap.h, 4358
- stl\_iterator.h, 4359, 4363
- stl\_iterator\_base\_funcs.h, 4363
- stl\_iterator\_base\_types.h, 4364
- stl\_list.h, 4365
- stl\_map.h, 4366
- stl\_multimap.h, 4367
- stl\_multiset.h, 4368
- stl\_numeric.h, 4369
- stl\_pair.h, 4370
- stl\_queue.h, 4371
- stl\_raw\_storage\_iter.h, 4372
- stl\_relops.h, 4372
- stl\_set.h, 4373
- stl\_stack.h, 4373
- stl\_tempbuf.h, 4374
- stl\_tree.h, 4375
- stl\_uninitialized.h, 4376
- stl\_vector.h, 4377
- str
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 2306
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2438
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2569
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2627
  - std::match\_results< \_Bi\_iter, \_Alloc >, 3263
  - std::sub\_match< \_Biter >, 3658
- stream\_iterator.h, 4378
- streambuf, 4379
  - I/O, 116
- streambuf.tcc, 4380
- streambuf\_iterator.h, 4380
- streambuf\_type
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 3139
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 3492
- streamoff
  - std, 671
- streampos
  - std, 671
- streamsize
  - std, 671
- stride
  - Numeric Arrays, 282
- string, 4381, 4384
  - Strings, 417
- string\_conversions.h, 4384
- string\_type
  - std::collate< \_CharT >, 2729
  - std::collate\_byname< \_CharT >, 2736
  - std::messages< \_CharT >, 3278
  - std::money\_get< \_CharT, \_InIter >, 3291
  - std::money\_put< \_CharT, \_OutIter >, 3297
  - std::moneypunct< \_CharT, \_Intl >, 3304
  - std::numpunct< \_CharT >, 3471
- string\_view, 4385
- string\_view.tcc, 4387
- stringbuf
  - I/O, 116

- stringfwd.h, [4387](#)
- Strings, [417](#)
  - string, [417](#)
  - u16string, [417](#)
  - u32string, [418](#)
  - wstring, [418](#)
- stringstream
  - I/O, [117](#)
- strstream, [4388](#)
- substr
  - \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>, [958](#)
  - std::basic\_string<\_CharT, \_Traits, \_Alloc>, [2550](#)
- subtract\_with\_carry\_engine
  - std::subtract\_with\_carry\_engine<\_UIntType, \_\_w, \_\_s, \_\_r>, [3661](#)
- subtractive\_rng
  - \_\_gnu\_cxx::subtractive\_rng, [1094](#)
- suffix
  - std::match\_results<\_Bi\_iter, \_Alloc>, [3264](#)
- sum
  - Numeric Arrays, [282](#)
- sungetc
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [1000](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [1064](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf<\_CharT, \_Traits>, [1088](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1985](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [2481](#)
  - std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>, [2570](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3951](#)
- swap
  - \_\_gnu\_cxx, [479](#)
  - \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>, [959](#)
  - \_\_gnu\_debug::basic\_string<\_CharT, \_Traits, \_Allocator>, [1199](#)
  - \_\_gnu\_pbds::sample\_probe\_fn, [1584](#)
  - \_\_gnu\_pbds::sample\_range\_hashing, [1586](#)
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, [1587](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [1592](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [1597](#)
  - \_\_gnu\_pbds::sample\_size\_policy, [1598](#)
  - \_\_gnu\_pbds::sample\_update\_policy, [1603](#)
  - experimental/bits/fs\_path.h, [4128](#)
  - Futures, [100](#)
  - Mutexes, [208](#)
  - Numeric Arrays, [282](#)
  - Regular Expressions, [371](#)
  - std, [779–783](#)
  - std::\_\_debug, [797](#)
  - std::\_\_profile, [824](#)
  - std::basic\_regex<\_Ch\_type, \_Rx\_traits>, [2462](#)
  - std::basic\_string<\_CharT, \_Traits, \_Alloc>, [2550](#)
  - std::deque<\_Tp, \_Alloc>, [2881](#)
  - std::experimental::fundamentals\_v1::any, [2913](#)
  - std::forward\_list<\_Tp, \_Alloc>, [2966](#)
  - std::function<\_Res(\_ArgTypes...)>, [2981](#)
  - std::list<\_Tp, \_Alloc>, [3184](#)
  - std::map<\_Key, \_Tp, \_Compare, \_Alloc>, [3247](#)
  - std::match\_results<\_Bi\_iter, \_Alloc>, [3264](#)
  - std::multimap<\_Key, \_Tp, \_Compare, \_Alloc>, [3356](#)
  - std::multiset<\_Key, \_Compare, \_Alloc>, [3384](#)
  - std::set<\_Key, \_Compare, \_Alloc>, [3614](#)
  - std::tr2::dynamic\_bitset<\_WordT, \_Alloc>, [3731](#)
  - std::unique\_ptr<\_Tp, \_Dp>, [3770](#)
  - std::unique\_ptr<\_Tp[], \_Dp>, [3777](#)
  - std::unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc>, [3810](#)
  - std::unordered\_multimap<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc>, [3840](#)
  - std::unordered\_multiset<\_Value, \_Hash, \_Pred, \_Alloc>, [3870](#)
  - std::unordered\_set<\_Value, \_Hash, \_Pred, \_Alloc>, [3899](#)
  - std::vector<\_Tp, \_Alloc>, [3930](#)
  - Type-safe container of any type, [428](#)
  - Utilities, [444, 445](#)
- swap\_ranges
  - Mutating, [202](#)
- sync
  - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [1000](#)
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [1064](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf<\_CharT, \_Traits>, [1089](#)
  - std::basic\_filebuf<\_CharT, \_Traits>, [1985](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [2046](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2101](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2197](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2251](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2306](#)
  - std::basic\_streambuf<\_CharT, \_Traits>, [2482](#)
  - std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>, [2570](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2627](#)
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, [3952](#)
- sync\_with\_stdio
  - std::basic\_fstream<\_CharT, \_Traits>, [2046](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [2101](#)
  - std::basic\_ios<\_CharT, \_Traits>, [2135](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [2197](#)
  - std::basic\_istream<\_CharT, \_Traits>, [2251](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [2307](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [2351](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [2393](#)

- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2439
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2628
- std::ios\_base, 3067
- syntax\_option\_type
  - std::regex\_constants, 841
- synth\_access\_traits
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 1535
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 1537
- synth\_access\_traits.hpp, 4388
- system\_error, 4388, 4389
- t
  - std::binomial\_distribution< \_IntType >, 2655
- table
  - std::ctype< char >, 2784
  - std::ctype\_byname< char >, 2837
- table\_size
  - std::ctype< char >, 2788
  - std::ctype\_byname< char >, 2841
- tag\_and\_trait.hpp, 4390
- Tags, 419
  - trivial\_iterator\_difference\_type, 419
- tags.h, 4391
- tan
  - Complex Numbers, 68
- tanh
  - Complex Numbers, 69
- target
  - std::function< \_Res(\_ArgTypes...) >, 2982
- target\_type
  - std::function< \_Res(\_ArgTypes...) >, 2982
- tellg
  - std::basic\_fstream< \_CharT, \_Traits >, 2047
  - std::basic\_ifstream< \_CharT, \_Traits >, 2102
  - std::basic\_iostream< \_CharT, \_Traits >, 2198
  - std::basic\_istream< \_CharT, \_Traits >, 2252
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2307
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2628
- tellp
  - std::basic\_fstream< \_CharT, \_Traits >, 2047
  - std::basic\_iostream< \_CharT, \_Traits >, 2198
  - std::basic\_ofstream< \_CharT, \_Traits >, 2351
  - std::basic\_ostream< \_CharT, \_Traits >, 2394
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2439
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2629
- temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 1096
- terminate
  - std, 784
- terminate\_handler
  - std, 671
- test
  - std::bitset< \_Nb >, 2671
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 3731
- tgmath.h, 4392
- thin\_heap.hpp, 4392
- thousands\_sep
  - std::moneypunct< \_CharT, \_Intl >, 3314
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3325
  - std::numpunct< \_CharT >, 3476
  - std::numpunct\_byname< \_CharT >, 3482
- thread, 4393
- Threads, 420
- throw\_allocator.h, 4394
- throw\_with\_nested
  - Exceptions, 87
- tie
  - std::basic\_fstream< \_CharT, \_Traits >, 2047, 2048
  - std::basic\_ifstream< \_CharT, \_Traits >, 2102, 2103
  - std::basic\_ios< \_CharT, \_Traits >, 2135, 2136
  - std::basic\_iostream< \_CharT, \_Traits >, 2198, 2199
  - std::basic\_istream< \_CharT, \_Traits >, 2252
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2308
  - std::basic\_ofstream< \_CharT, \_Traits >, 2352
  - std::basic\_ostream< \_CharT, \_Traits >, 2394
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 2439, 2440
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2629
  - Utilities, 445
- Time, 421
- time
  - std::locale, 3197
- time\_get
  - std::time\_get< \_CharT, \_Intl >, 3671
- time\_members.h, 4395
- time\_point\_cast
  - std::chrono, 828
- time\_put
  - std::time\_put< \_CharT, \_Intl >, 3698
- tinyness\_before
  - std::\_\_numeric\_limits\_base, 1828
  - std::numeric\_limits< \_Tp >, 3450
- TLB\_size
  - \_\_gnu\_parallel::Settings, 1315



- to\_bytes
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_al-  
loc, \_Byte\_alloc >, [3965](#), [3966](#)
- to\_string
  - std::bitset< \_Nb >, [2672](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3732](#)
- to\_ullong
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3732](#)
- to\_ulong
  - std::bitset< \_Nb >, [2672](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [3732](#)
- tolower
  - std, [784](#)
  - std::\_\_ctype\_abstract\_base< \_CharT >, [1693](#), [1694](#)
  - std::ctype< \_CharT >, [2767](#)
  - std::ctype< char >, [2785](#)
  - std::ctype< wchar\_t >, [2803](#)
  - std::ctype\_byname< \_CharT >, [2820](#)
  - std::ctype\_byname< char >, [2837](#), [2838](#)
- top
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3535](#)
  - std::stack< \_Tp, \_Sequence >, [3649](#)
- toupper
  - std, [784](#)
  - std::\_\_ctype\_abstract\_base< \_CharT >, [1695](#)
  - std::ctype< \_CharT >, [2769](#)
  - std::ctype< char >, [2786](#)
  - std::ctype< wchar\_t >, [2804](#)
  - std::ctype\_byname< \_CharT >, [2822](#)
  - std::ctype\_byname< char >, [2838](#), [2839](#)
- trace\_fn\_imps.hpp, [4395](#), [4396](#)
- Traits, [422](#)
- traits.hpp, [4397](#)–[4399](#)
- traits\_type
  - std::basic\_ios< \_CharT, \_Traits >, [2121](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [2263](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2468](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [3139](#)
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3487](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3492](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3938](#)
- transform
  - Mutating, [203](#)
  - std::collate< \_CharT >, [2733](#)
  - std::collate\_byname< \_CharT >, [2739](#)
  - std::regex\_traits< \_Ch\_type >, [3569](#)
- transform\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1315](#)
- transform\_primary
  - std::regex\_traits< \_Ch\_type >, [3570](#)
- translate
  - std::regex\_traits< \_Ch\_type >, [3570](#)
- translate\_nocase
  - std::regex\_traits< \_Ch\_type >, [3571](#)
- traps
  - std::\_\_numeric\_limits\_base, [1828](#)
  - std::numeric\_limits< \_Tp >, [3450](#)
- tree
  - \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag,  
Node\_Update, \_Alloc >, [1609](#)
- tree\_policy.hpp, [4399](#)
- tree\_trace\_base.hpp, [4399](#)
- trie
  - \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag,  
Node\_Update, \_Alloc >, [1615](#)
- trie\_policy.hpp, [4400](#)
- trie\_policy\_base.hpp, [4400](#)
- trie\_string\_access\_traits\_imp.hpp, [4401](#)
- trivial\_iterator\_difference\_type
  - Tags, [419](#)
- true\_type
  - Metaprogramming, [181](#)
- trunename
  - std::num\_punct< \_CharT >, [3476](#)
  - std::num\_punct\_byname< \_CharT >, [3482](#)
- trunc
  - std::basic\_fstream< \_CharT, \_Traits >, [2058](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2113](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2145](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2209](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2262](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2318](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2362](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2404](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2449](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2640](#)
  - std::ios\_base, [3075](#)
- try\_lock
  - std, [784](#)
- try\_to\_lock
  - Mutexes, [209](#)
- tuple, [4401](#), [4403](#)
- tuple\_cat
  - Utilities, [445](#)
- type
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,  
Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >, [1411](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,  
Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >, [1412](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key,  
Mapped, \_Alloc, list\_update\_tag, Policy\_TI >, [1413](#)

- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_Tl >, [1413](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_Tl >, [1414](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_Tl >, [1415](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_Tl >, [1416](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_Tl >, [1417](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_Tl >, [1418](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_Tl >, [1418](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_Tl >, [1419](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_Tl >, [1420](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, [1407](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, [1408](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, [1409](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, [1410](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, [1410](#)
- \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [1421](#)
- \_\_gnu\_pbds::detail::default\_eq\_fn< Key >, [1422](#)
- \_\_gnu\_pbds::detail::default\_hash\_fn< Key >, [1422](#)
- \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn >, [1423](#)
- \_\_gnu\_pbds::detail::default\_resize\_policy< Comb\_Hash\_Fn >, [1424](#)
- \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string< Char, Char\_Traits, std::allocator< Char > >, [1425](#)
- \_\_gnu\_pbds::detail::default\_update\_policy, [1425](#)
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, [1428](#)
- std::aligned\_union< \_Len, \_Types >, [1903](#)
- std::experimental::fundamentals\_v1::any, [2913](#)
- Type-safe container of any type, [424](#)
- any\_cast, [425–427](#)
- swap, [428](#)
- type\_traits, [4404](#), [4408](#), [4409](#)
- \_GLIBCXX\_HAS\_NESTED\_TYPE, [4408](#)
- type\_traits.h, [4412](#)
- type\_utils.hpp, [4413](#)
- typeid, [4413](#)
- typeid, [4414](#)
- typelist.h, [4414](#)
- types.h, [4416](#)
- types\_traits.hpp, [4416](#)
- u16streampos
  - std, [671](#)
- u16string
  - Strings, [417](#)
- u32streampos
  - std, [672](#)
- u32string
  - Strings, [418](#)
- u8path
  - experimental/bits/fs\_path.h, [4128](#)
- uflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [1000](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1064](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1089](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1985](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2482](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2570](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3952](#)
- uncaught\_exception
  - std, [785](#)
- uncaught\_exceptions
  - std, [785](#)
- underflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [1001](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1065](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1089](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1986](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2482](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2571](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3953](#)
- underlying\_type\_t
  - Metaprogramming, [181](#)
- unexpected
  - std, [785](#)
- unexpected\_handler
  - std, [672](#)
- unset
  - std::basic\_fstream< \_CharT, \_Traits >, [2048](#)

- std::basic\_ifstream< \_CharT, \_Traits >, [2103](#)
- std::basic\_iostream< \_CharT, \_Traits >, [2199](#)
- std::basic\_istream< \_CharT, \_Traits >, [2253](#)
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2309](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2630](#)
- Uniform Distributions, [429](#)
  - operator!=, [429](#), [430](#)
  - operator<<, [430](#)
  - operator>>, [431](#)
- uniform\_int\_dist.h, [4417](#)
- uniform\_int\_distribution
  - std::uniform\_int\_distribution< \_IntType >, [3755](#)
- uniform\_real\_distribution
  - std::uniform\_real\_distribution< \_RealType >, [3759](#)
- uninitialized\_copy
  - std, [785](#)
- uninitialized\_copy\_n
  - SGL, [382](#)
  - std, [786](#)
- uninitialized\_fill
  - std, [786](#)
- uninitialized\_fill\_n
  - std, [787](#)
- unique
  - Mutating, [204](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2966](#)
  - std::list< \_Tp, \_Alloc >, [3184](#)
- unique\_copy
  - Mutating, [205](#), [206](#)
- unique\_copy.h, [4417](#)
- unique\_copy\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1316](#)
- unique\_ptr
  - std::unique\_ptr< \_Tp, \_Dp >, [3764–3766](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3771–3773](#)
- unique\_ptr.h, [4418](#)
- unitbuf
  - std, [787](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2059](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2113](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2145](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2210](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2262](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2318](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2362](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2404](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2450](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2640](#)
  - std::ios\_base, [3075](#)
- Unordered Associative, [433](#)
- unordered\_base.h, [4419](#)
- unordered\_map, [4420–4422](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3785–3787](#)
- unordered\_map.h, [4422](#)
- unordered\_multimap
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3818–3820](#)
- unordered\_multiset
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3847–3850](#)
- unordered\_set, [4424–4426](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3877–3880](#)
- unordered\_set.h, [4426](#)
- unsetf
  - std::basic\_fstream< \_CharT, \_Traits >, [2049](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2104](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2136](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2200](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2253](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2309](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2352](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2395](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2440](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2630](#)
  - std::ios\_base, [3067](#)
- unshift
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, [1679](#)
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, [2695](#)
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, [2700](#)
  - std::codecvt< char, char, mbstate\_t >, [2705](#)
  - std::codecvt< char16\_t, char, mbstate\_t >, [2710](#)
  - std::codecvt< char32\_t, char, mbstate\_t >, [2715](#)
  - std::codecvt< wchar\_t, char, mbstate\_t >, [2720](#)
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, [2726](#)
- update\_fn\_imps.hpp, [4427](#)
- upper\_bound
  - Binary Search, [45](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [3247–3249](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3356–3358](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3385](#), [3386](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3614–3616](#)
- uppercase
  - std, [788](#)

- std::basic\_fstream< \_CharT, \_Traits >, 2059
- std::basic\_ifstream< \_CharT, \_Traits >, 2113
- std::basic\_ios< \_CharT, \_Traits >, 2145
- std::basic\_iostream< \_CharT, \_Traits >, 2210
- std::basic\_istream< \_CharT, \_Traits >, 2262
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 2318
- std::basic\_ofstream< \_CharT, \_Traits >, 2362
- std::basic\_ostream< \_CharT, \_Traits >, 2404
- std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 2450
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2640
- std::ios\_base, 3075
- use\_facet
  - Locales, 131
  - std::locale, 3194
  - std::locale::id, 3201
- Utilities, 434
  - \_\_addressof, 437
  - \_\_invoke, 437
  - addressof, 438
  - forward, 438
  - get, 439, 440
  - make\_pair, 441
  - move, 441
  - move\_if\_noexcept, 442
  - operator!=, 442
  - operator<, 442
  - operator<=, 443
  - operator>, 443
  - operator>=, 443
  - operator==, 443
  - piecewise\_construct, 446
  - swap, 444, 445
  - tie, 445
  - tuple\_cat, 445
- utility, 4428, 4429
- valarray, 4430
  - Numeric Arrays, 248–250
  - std::valarray< \_Tp >, 3903
- valarray\_after.h, 4434
- valarray\_array.h, 4443
- valarray\_array.tcc, 4451
- valarray\_before.h, 4452
- valid\_prefix
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 1485
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 1489
- value
  - std::regex\_traits< \_Ch\_type >, 3571
- value\_comp
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 3249
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3358
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3386
  - std::set< \_Key, \_Compare, \_Alloc >, 3616
- value\_compare
  - std::set< \_Key, \_Compare, \_Alloc >, 3593
- value\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1372
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 1378
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1387
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 1392
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 1443
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, 1449
- const\_iterator\_, 1649
- iterator\_, 1655
- point\_const\_iterator\_, 1660
- point\_iterator\_, 1665
- std::allocator\_traits< \_Alloc >, 1912
- std::allocator\_traits< allocator< \_Tp > >, 1919
- std::back\_insert\_iterator< \_Container >, 1956
- std::complex< \_Tp >, 2742
- std::front\_insert\_iterator< \_Container >, 2972
- std::insert\_iterator< \_Container >, 3051
- std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 3135
- std::istreambuf\_iterator< \_CharT, \_Traits >, 3139
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 3143
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3488
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 3492
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 3551
- std::reverse\_iterator< \_Iterator >, 3577
- std::set< \_Key, \_Compare, \_Alloc >, 3593
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3785
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3818
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3847
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3877
- vector, 4453, 4454
  - std::\_\_debug::vector< \_Tp, \_Allocator >, 1762
  - std::vector< \_Tp, \_Alloc >, 3907–3910

- vector.tcc, [4455](#)
- void\_pointer
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [868](#)
  - std::allocator\_traits< \_Alloc >, [1912](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1919](#)
- void\_t
  - Metaprogramming, [181](#)
- vstring.h, [4455](#)
- vstring.tcc, [4458](#)
- vstring\_fwd.h, [4459](#)
- vstring\_util.h, [4459](#)
- wbuffer\_convert
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3938](#)
- wcerr
  - std, [791](#)
- wcin
  - std, [791](#)
- wclog
  - std, [791](#)
- wcout
  - std, [791](#)
- wcregex\_token\_iterator
  - Regular Expressions, [332](#)
- wcsub\_match
  - Regular Expressions, [332](#)
- wfilebuf
  - I/O, [117](#)
- wfstream
  - I/O, [117](#)
- what
  - \_\_gnu\_pbds::container\_error, [1360](#)
  - \_\_gnu\_pbds::insert\_error, [1562](#)
  - \_\_gnu\_pbds::join\_error, [1564](#)
  - \_\_gnu\_pbds::resize\_error, [1582](#)
  - Exceptions, [87](#)
  - std::bad\_alloc, [1958](#)
  - std::bad\_cast, [1960](#)
  - std::bad\_exception, [1961](#)
  - std::bad\_function\_call, [1962](#)
  - std::bad\_typeid, [1963](#)
  - std::bad\_weak\_ptr, [1964](#)
  - std::domain\_error, [2898](#)
  - std::experimental::fundamentals\_v1::bad\_any\_cast, [2915](#)
  - std::experimental::fundamentals\_v1::bad\_optional\_access, [2916](#)
  - std::future\_error, [2992](#)
  - std::invalid\_argument, [3056](#)
  - std::ios\_base::failure, [3077](#)
  - std::length\_error, [3145](#)
  - std::logic\_error, [3204](#)
  - std::out\_of\_range, [3495](#)
  - std::overflow\_error, [3497](#)
  - std::range\_error, [3545](#)
  - std::regex\_error, [3555](#)
  - std::runtime\_error, [3583](#)
  - std::system\_error, [3666](#)
  - std::underflow\_error, [3753](#)
- widen
  - std::\_\_ctype\_abstract\_base< \_CharT >, [1696](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [2049](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2104](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2137](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2200](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2254](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2310](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2353](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2395](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2441](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2631](#)
  - std::ctype< \_CharT >, [2770](#)
  - std::ctype< char >, [2787](#), [2788](#)
  - std::ctype< wchar\_t >, [2805](#)
  - std::ctype\_byname< \_CharT >, [2823](#)
  - std::ctype\_byname< char >, [2840](#)
- width
  - std::basic\_fstream< \_CharT, \_Traits >, [2050](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2105](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2137](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2201](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2254](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2310](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2353](#), [2354](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2396](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [2441](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2631](#)
  - std::ios\_base, [3067](#), [3068](#)
- wfstream
  - I/O, [117](#)
- wios
  - I/O, [117](#)
- wiostream
  - I/O, [118](#)
- wistream
  - I/O, [118](#)
- wistringstream
  - I/O, [118](#)
- wofstream
  - I/O, [118](#)
- workstealing.h, [4460](#)
- wostream

- I/O, [118](#)
- wostringstream
  - I/O, [119](#)
- wregex
  - Regular Expressions, [333](#)
- write
  - std::basic\_fstream< \_CharT, \_Traits >, [2051](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2202](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2354](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2396](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2442](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2632](#)
- ws
  - std, [788](#)
- wsregex\_token\_iterator
  - Regular Expressions, [333](#)
- wssub\_match
  - Regular Expressions, [333](#)
- wstreambuf
  - I/O, [119](#)
- wstreampos
  - std, [672](#)
- wstring
  - Strings, [418](#)
- wstring\_convert
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, [3962](#), [3963](#)
- wstringbuf
  - I/O, [119](#)
- wstringstream
  - I/O, [119](#)
- xalloc
  - std::basic\_fstream< \_CharT, \_Traits >, [2051](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [2106](#)
  - std::basic\_ios< \_CharT, \_Traits >, [2138](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [2202](#)
  - std::basic\_istream< \_CharT, \_Traits >, [2255](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [2311](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [2355](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [2397](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [2442](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2632](#)
  - std::ios\_base, [3068](#)
- xsgen
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [1001](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1065](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1090](#)
- std::basic\_filebuf< \_CharT, \_Traits >, [1986](#)
- std::basic\_streambuf< \_CharT, \_Traits >, [2483](#)
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2571](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3953](#)
- xspn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [1002](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [1066](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [1091](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1987](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [2484](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2572](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3954](#)
- yield
  - std::this\_thread, [858](#)